Defense Nuclear Agency
Alexandria, VA 22310-3398

DNA-TR-90-81

BETAFACT: A Code for the Statistical Analysis of Algorithms

APTEK, Inc.
1257 Lake Plaza Drive
Colorado Springs, CO 80906-3578

October 1990

Technical Report

CONTRACT No. DNA 001-88-C-0040

Destroy this report when it is no longer needed. Do not return to sender.

PLEASE NOTIFY THE DEFENSE NUCLEAR AGENCY, ATTN: CSTI, 6801 TELEGRAPH ROAD, ALEXANDRIA, VA 22310-3398, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH IT DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

DISTRIBUTION LIST UPDATE

This mailer is provided to enable DNA to maintain current distribution lists for reports. We would appreciate your providing the requested information.

NOTE:
Please return the mailing label from the document so that any additions, changes, corrections or deletions can be made more easily.

☐ Add the individual listed to your distribution list.

☐ Delete the cited organization/individual.

☐ Change of address.

NAME: _____

ORGANIZATION: _____

OLD ADDRESS	CURRENT ADDRESS
_____	_____
_____	_____
_____	_____

TELEPHONE NUMBER: ( ) _____

SUBJECT AREA(s) OF INTEREST:

_____	_____
_____	_____
_____	_____

DNA OR OTHER GOVERNMENT CONTRACT NUMBER: _____

CERTIFICATION OF NEED-TO-KNOW BY GOVERNMENT SPONSOR (if other than DNA):

SPONSORING ORGANIZATION: _____

CONTRACTING OFFICER OR REPRESENTATIVE: _____

SIGNATURE: _____

CUT HERE AND RETURN

Director
Defense Nuclear Agency
ATTN. TITL
Washington, DC 20305-1000

Director
Defense Nuclear Agency
ATTN: TITL
Washington, DC 20305-1000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	901001	Technical - 890228 to 900228

4. TITLE AND SUBTITLE
BETAFACT: A Code for the Statistical Analysis of Algorithms

5. FUNDING NUMBERS
C - DNA001-88-C-0040
PE - 62715H
PR - X
TA - C
WU - 047280

6. AUTHOR(S)
Stephen H. Sutherland

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
APTEK, Inc.
1257 Lake Plaza Drive
Colorado Springs, CO 80906-3578

8. PERFORMING ORGANIZATION REPORT NUMBER
A-90-1R

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398
SPWE/Wolf

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
DNA-TR-90-81

11. SUPPLEMENTARY NOTES
This work was sponsored by the Defense Nuclear Agency under RDT&E RMC Codes B342085764 X C 00017 25904D and B342085764 X C 00018 25904D.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In the course of performing lethality assessments for the Defense Nuclear Agency Single Pulse Laser Lethality and Target Hardening (LTH-3) Program, it is necessary to consistently combine many sources of uncertainty which contribute to the overall uncertainty associated with algorithms used in modelling laser interaction and target response. A computer code called BETAFACT was developed to perform this task.

A Monte Carlo approach is used in BETAFACT to develop an algorithm results distribution which reflects the combined effect of all sources of uncertainty which contribute to the overall algorithm uncertainty. Statistics are evaluated for the algorithm results distribution to quantify the overall associated uncertainty and to identify a probability distribution which models the algorithm results distribution. Normal, lognormal, Beta, and generalized uniform distributions are used in the code to model uncertainties. The code runs interactively, prompting for data and analysis modifications as required. It is necessary for the user of the code to have a very modest proficiency in FORTRAN programming in order to make a specific algorithm accessible to the code via user provided subroutines.

14. SUBJECT TERMS		15. NUMBER OF PAGES
Lethality Assessment	K-Factors	104
Algorithms	Monte Carlo	16. PRICE CODE
Uncertainty	LTH-3	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

CLASSIFIED BY:

N/A since Unclassified

DECLASSIFY ON:
N/A since Unclassified

SUMMARY

At APTEK, Inc. we use an algorithm based probabilistic methodology to perform lethality assessments for the Defense Nuclear Agency Single Pulse Laser Lethality and Target Hardening (LTH-3) Program. The algorithms utilized relate the response of laser illuminated targets to local effects (such as delivered impulse) induced by the impinging laser radiation. In general the algorithms are functions of laser parameters (eg., fluence and pulse duration) and target design details (eg., physical dimensions, construction material properties, pressurization levels, etc.) Typically the correlation between algorithm predictions and relevant experimental data is at best fair, and it is always the case that target design details are imprecisely known since we are dealing with foreign targets. Thus uncertainty is associated with our algorithm predictions because of the uncertainty associated with algorithm parameters compounded with the uncertainty attributable to the algorithms themselves. The goal of the work described in this report was to develop and implement a numerical approach for consistently compounding all sources of uncertainty in order to obtain a quantified estimate of the overall uncertainty which should be associated with an algorithm and the predictions obtained with it. The combined uncertainty estimates are vital inputs to our lethality assessment activities.

When one is attempting to consistently combine contributing sources of error or uncertainty to obtain an overall uncertainty estimate, it is necessary to consider several important factors. Among these factors are the characteristics of the probability distribution associated with each source of uncertainty, possible correlation between different sources of uncertainty, the eventual use of the combined uncertainty estimate, and practical aspects of implementing a procedure to obtain the combined uncertainty estimates. To a varying extent all of these factors, and others besides, influenced the development of the finished product of this effort, the BETAFACT code.

BETAFACT is a Monte Carlo based code which enables the user to numerically estimate the overall uncertainty associated with an arbitrary algorithm which has an arbitrary number of contributing sources of uncertainty. Normal, lognormal, Beta and generalized uniform probability distributions, as selected by the analyst, are used in the code to model contributing uncertainties. At present, for a given analysis all uncertainty sources must be modeled with the same type of probability distribution (i.e., all normal or all Beta distributed) and sources of uncertainty must be uncorrelated. Since the code runs interactively, once an algorithm is defined it is an easy matter to reanalyze the algorithm for different probability distributions and uncertainty level specifications. The latter capability enables the user to perform sensitivity studies which means that the code is also a useful decision making and program management tool.

In this report we describe the technical background of the approach used to combine uncertainty sources, discuss implementation of the theory in BETAFACT,

present illustrative examples of the use of the code, and provide a user's manual. Together these will rapidly acquaint a potential user with the operation of the code.

CONVERSION TABLE

Conversion factors for U.S. Customary to metric (SI) units of measurement.

MULTIPLY ⟶ TO GET ⟵	⟶ BY ⟶ ⟵ BY ⟵	⟶ TO GET ⟵ DIVIDE
angstrom	1.000 000 × E -10	meters (m)
atmosphere (normal)	1.013 25 × E +2	kilo pascal (kPa)
bar	1.000 000 × E +2	kilo pascal (kPa)
barn	1.000 000 × E -28	meter^2 (m^2)
British thermal unit (thermochemical)	1.054 350 × E +3	joule (J)
calorie (thermochemical)	4.184 000	joule (J)
cal (thermochemical)/cm^2	4.184 000 × E -2	mega joule/m^2 (MJ/m^2)
curie	3.700 000 × E +1	giga becquerel (GBq)*
degree (angle)	1.745 329 × E -2	radian (rad)
degree Fahrenheit	$\tau_K = (t°f + 459.67)/1.8$	degree kelvin (K)
electron volt	1.602 19 × E -19	joule (J)
erg	1.000 000 × E -7	joule (J)
erg/second	1.000 000 × E -7	watt (W)
foot	3.048 000 × E -1	meter (m)
foot-pound-force	1.355 818	joule (J)
gallon (U.S. liquid)	3.785 412 × E -3	meter^3 (m^3)
inch	2.540 000 × E -2	meter (m)
jerk	1.000 000 × E +9	joule (J)
joule/kilogram (J/kg) (radiation dose absorbed)	1.000 000	Gray (Gy) **
kilotons	4.183	terajoules
kip (1000 lbf)	4.448 222 × E +3	newton (N)
kip/inch^2 (ksi)	6.894 757 × E +3	kilo pascal (kPa)
ktap	1.000 000 × E +2	newton- second/m^2 (N-s/m^2)
micron	1.000 000 × E -6	meter (m)
mil	2.540 000 × E -5	meter (m)
mile (international)	1.609 344 × E +3	meter (m)
ounce	2.834 952 × E -2	kilogram (kg)
pound-force (lbf avoirdupois)	4.448 222	newton (N)
pound-force inch	1.129 848 × E -1	newton-meter (N•m)
pound-force inch	1.751 268 × E +2	newton/meter (N/m)
pound-force/foot^2	4.788 026 × E -2	kilo pascal (kPa)
pound-force/inch^2 (psi)	6.894 757	kilo pascal (kPa)
pound-mass (lbm avoirdupois)	4.535 924 × E -1	kilogram (kg)
pound-mass-foot^2 (moment of inertia)	4.214 011 × E -2	kilogram-meter^2 (kg•m^2)
pound-mass/foot^3	1.601 846 × E +1	kilogram/meter^3 (kg/m^3
rad (radiation dose absorbed)	1.000 000 × E -2	Gray (Gy)**
roentgen	2.579 760 × E -4	coulomb/kilogram (C/kg)
shake	1.000 000 x E -8	second (s)
slug	1.459 390 x E +1	kilogram (kg)
torr (mm Hg, 0°C)	1.333 22 × E -1	kilo pascal (kPa)

* The becquerel (Bq) is the SI unit of radioactivity; 1 Bq = 1 event/s.

**The Gray (Gy) is the SI unit of absorbed radiation.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

SECTION 1
INTRODUCTION

The BETAFACT code was developed by APTEK to support the preparation of yearly lethality assessments (APTEK, 1988 and APTEK, 1990) for the Defense Nuclear Agency (DNA) Single Pulse Laser Lethality and Target Hardening (LTH-3) Program. The results reported in these lethality assessments were generated using an APTEK modified version of the DNA FAST (Failure Analysis by Statistical Techniques) code (Rowan, 1974). Among its several uses, BETAFACT is utilized to generate input required in the modelling and execution of FAST analyses.

The numerical models which can be analyzed with FAST are defined in terms of mathematical relations called algorithms. A typical algorithm consists of an output quantity which is a function of several input parameters. In the usual case, each of the input parameters does not have a precisely known value but does have a most probable (or nominal) value and a distribution of possible values about the nominal. Such a parameter is termed a parameter with associated uncertainty in this report. It is usually possible (besides being convenient) to characterize or approximate the distribution of possible values for a parameter with associated uncertainty in the form of standard probability distributions such as the normal and the uniform distributions.

Clearly, if an algorithm is a function of one or more parameters with associated uncertainty, the output of the algorithm must also have associated uncertainty and a distribution of possible results. The output uncertainty and output distribution are functions of the input parameter uncertainties and distributions and the details of the algorithm. A primary function of BETAFACT is to properly combine the input parameter uncertainties and distributions in order to determine the uncertainty which should be associated with the algorithm output.

An algorithm, viewed from an overall perspective, may also have an associated uncertainty, regardless of whether or not its input parameters have associated uncertainty. This situation can arise, for instance, if we know the values of input parameters accurately (to within a few percent) while the correlation between the parameters provided by the algorithm is only known approximately (e.g., within plus or minus 50%). In this situation, the input parameter uncertainties could be viewed as ignorable, and the uncertainty associated with the algorithm output would be that applicable to the algorithm itself. However, if the input parameter uncertainties are comparable in magnitude to the algorithm uncertainty or there are several input parameters with smaller but still not ignorable associated uncertainty, then all of these sources of uncertainty have to be combined to obtain the uncertainty which should be associated with the algorithm output. The BETAFACT code is capable of handling both of these situations.

Since the code combines several sources of uncertainty into an overall uncertainty associated with an algorithm output, BETAFACT can also be used to perform various types of sensitivity studies. For instance, suppose one has an algorithm of in-

terest which has two input parameters, each with significant associated uncertainty. By varying the uncertainty specified with each of the two parameters and observing the effect of the variations on the overall combined uncertainty estimated with BETAFACT, the more significant of the two uncertainty sources in producing the overall uncertainty can be identified. While such a procedure may not be necessary in order to recognize the more significant uncertainty source in a simple algorithm, recognition of that source can be much more difficult in a complicated algorithm. As another example, again suppose an algorithm uses two parameters, each of which has associated uncertainty and is obtained by experiment. A project manager or experimenter could use BETAFACT to evaluate the benefit of directing program resources to reduction of the parameter uncertainties both in terms of resources expended and reduction in overall uncertainty of the algorithm. In a later section of this report we will provide examples illustrative of the above uses of BETAFACT.

The above overview provides a brief description of the function of BETAFACT and some simple examples of how the program can be used. In the following report section, we present background material on algorithms and treatment of their sources of uncertainty and review some properties of the four types of probability distributions (normal, lognormal, Beta, and uniform) which are available in BETAFACT for modelling parameter and algorithm uncertainties.

The manner in which algorithms, probability distributions, and uncertainties are handled numerically in BETAFACT is the subject of Section 3. This section is particularly important to the user of the code since it defines the K-factor method we employ to represent uncertainty specifications. In Section 4 we consider several example problems which illustrate the use of BETAFACT and verify, in part, that the theory presented in Section 2 is properly implemented in the code. This section also addresses the method which must be used to get a user's algorithm into the code.

Section 5 of this report is the User's Manual for BETAFACT. Since the code runs in an interactive mode, prompting for input as required, the user will likely find little need for the User's Manual after the program has once been used successfully. In Section 5 we will describe the actual steps required to setup and run the code and explain each of the prompts delivered to the user by BETAFACT during execution. We will then make some brief concluding remarks in Section 6. Finally, listings of example problem algorithms are given in Appendix A and a complete listing of BETAFACT is presented in Appendix B.

Before proceeding to Section 2, we make note that BETAFACT was developed on a MicroVaxII computer operating under VMS. The code is written in standard FORTRAN-77 and is nearly entirely flexibly dimensioned. Although not verified, it is likely the program will also execute without modification on personal computers. Finally we mention that the user of BETAFACT must have simple programming experience with FORTRAN in order to use the code. This is a requirement because the algorithm of interest to a user must be coded in a FORTRAN subroutine so that it will be available to BETAFACT.

SECTION 2
BACKGROUND ON ALGORITHMS AND PROBABILITY DISTRIBUTIONS

The primary function of BETAFACT is to determine the overall uncertainty associated with the output of an algorithm, given the uncertainties associated with the algorithm and/or its input parameters and corresponding probability distribution specifications. The code performs this function by numerically evaluating the algorithm many times using probabilistically distributed values for its input parameters, applying (if necessary) algorithm uncertainty to the result of each evaluation, and computing statistics of the results distribution to quantify the uncertainty associated with the output. This can be done for an algorithm with essentially any functional form as long as the sources of uncertainty associated with the algorithm can be modelled adequately using either normal, lognormal, Beta, or uniform probability distributions. The present version of the code does not allow different sources of uncertainty in an algorithm to be modelled with different types of probability distributions. All uncertainties associated with an algorithm must be modelled as normally distributed or Beta distributed, etc. In our applications, we have not found it necessary to mix probability distribution types, and the code reflects this experience.

The purpose of the first subsection (2.1) of this report section is to provide additional information about what we mean by the concept of an algorithm, to describe a simple method for quantifying the overall uncertainty associated with an algorithm, and to give an explanation of why numerical treatment of algorithm output uncertainty is desirable (and in some cases imperative). In subsection 2.2 we then present some details concerning the probability distributions used in BETAFACT and relevant characteristics of the distributions.

2.1 ALGORITHMS AND UNCERTAINTIES.

In the present context, an algorithm is a mathematical function which relates an output or response (r) to one or more input parameters (x_i, i = 1,..., number of parameters N). Symbolically we write

$$r = r(x_1, x_2, \ldots, x_N) \tag{1}$$

in which the left hand side is the output of the algorithm and the right hand side represents the functional form of the algorithm. There is really no restriction on the form of the function on the right hand side of Equation 1 except that the output r must not be an argument in the function (i.e., r as a variable must not appear in both sides of the equation) and the function should be well-behaved (i.e. finitely bounded) over the full range of response values anticipated. Otherwise, the function $r(x_i)$ can be polynomial, exponential, sinusoidal, etc., and involve any combinations of these functional types.

A specific example of a simple algorithm is the elementary relation between the peak bending stress σ in a straight beam, the applied bending moment M, the area moment of inertia I of the beam cross section, and the extreme fiber distance c from the beam neutral axis.

$$\sigma = \frac{Mc}{I} \tag{2}$$

In this form, this relation is an algorithm for the peak bending stress σ. Given M, c, and I, we can predict the peak stress in the beam. However, we could also solve this equation for M in terms of σ, I, and c to obtain an algorithm for M. If we accept the assumption that the formula in Equation 2 is exact, the output of the algorithm would be uncertain only if some of the input parameters had associated uncertainty.

As an example of a somewhat more complicated algorithm, we consider one often used in the LTH-3 Program; the so-called HKL algorithm. One version of this algorithm is

$$\varepsilon = \frac{I^2}{2\rho\sigma h^2} \tag{3}$$

It is used to predict the peak axial strain ε which results in a metallic cylinder of thickness h, with wall material density ρ and ultimate strength σ, which is loaded on its side with a cosine distributed impulse of peak intensity I. This algorithm is derived based on theoretical considerations. Given good (i.e., known or measured with low associated uncertainty) values for the parameters on the right hand side of the equation, the strain predictions obtained with Equation 3 have been shown to agree to within about $\pm 50\%$ with experimentally measured strains. It is thus an example of an algorithm which has an associated algorithmic uncertainty. If we used this algorithm to predict the axial strain resulting in a cylinder whose ρ and σ values were inaccurately known but we knew h and I with considerable precision, we would have the situation of an algorithm with uncertainty associated with two input parameters and the algorithm itself. All three sources of uncertainty contribute to the uncertainty associated with the strain estimate for that cylinder.

Given an algorithm of the form of Equation 1, a simple standard method is available for estimating the uncertainty associated with the algorithm output if one or more of its input parameters has associated uncertainty. First, let U_i represent the uncertainty associated with the i-th parameter (x_i) in the algorithm. We assume that U_i is not correlated with the uncertainty U_j associated with the j-th parameter for all pairs i and j. (In this report and in the BETAFACT code, the uncertainties associated with algorithm input parameters are all assumed to be uncorrelated.) Typically U_i is selected or specified such that 95% of the possible values of x_i fall in the range associated with U_i about the nominal (or 50% or best estimate) value for x_i. (Relating the nominal value of x_i, U_i and the applicable range for x_i are considered in greater detail in subsection 2.2.) Now if only the i-th parameter of the algorithm is assumed to have associated uncertainty, then the corresponding uncertainty U_{ri} in the algorithm output can be evaluated as

$$U_{ri} = \frac{\partial r}{\partial x_i} U_i \tag{4}$$

The usual situation is that several input parameters in the algorithm have associated uncertainty. Equation 4 can be evaluated for each of these uncertainty sources considered separately, but the question then becomes one of how to combine the individual contributions to obtain a reasonable estimate for the algorithm output uncertainty. A recommended combination rule (Coleman and Steele, 1989) is to use the root-sum-square (RSS) of the U_{ri}. Thus, if we represent the combined output uncertainty as U_r, the RSS expression for U_r in terms of the U_{ri} is

$$U_r^2 = \sum_{i=1}^{N} U_{ri}^2 = \sum_{i=1}^{N} \left[\frac{\partial r}{\partial x_i} U_i \right]^2 \tag{5}$$

Given the algorithm r and the parameter uncertainty specifications U_i, Equation 5 is easily evaluated to yield U_r.

It is sometimes found that a more convenient form of Equation 5 is to divide both sides of it by r^2 (i.e., the algorithm squared) to obtain

$$\left(\frac{U_r}{r} \right)^2 = \sum_{i=1}^{N} \left[\frac{1}{r} \frac{\partial r}{\partial x_i} U_i \right]^2 \tag{6}$$

This equation often will give algebraically simpler expressions for U_r^2 than will Equation 5.

Although the above results are straightforward, it is perhaps useful to apply them to consideration of an example algorithm; a hypothetical one in this case. Suppose $r = A^2B/C$ for which the nominal parameter values are A = 10, B = 20, and C = 50 and the associated uncertainties are $U_A = \pm 2, U_B = \pm 4$, and $U_C = \pm 15$. The nominal output of the algorithm is 40. Differentiating the algorithm, we find that

$$\frac{1}{r} \frac{\partial r}{\partial A} U_A = 2 \frac{U_A}{A} \quad \frac{1}{r} \frac{\partial r}{\partial B} U_B = \frac{U_B}{B} \quad \frac{1}{r} \frac{\partial r}{\partial C} U_C = \frac{U_C}{C} \tag{7}$$

and thus

$$U_r^2 = \left[4 \left(\frac{U_A}{A} \right)^2 + \left(\frac{U_B}{B} \right)^2 + \left(\frac{U_C}{C} \right)^2 \right] r^2 \tag{8}$$

Substituting for the given values we find $U_r = \pm 21.5$. If we set $U_B = U_C = 0$, then we find that $U_r = \pm 16$ which shows that U_A accounts for the most significant portion of the combined uncertainty even though the uncertainty associated with $A(\pm 20\%)$ is the same as that associated with B, but less than that associated with $C(\pm 30\%)$. The exponent of A in the algorithm, which has an absolute value greater than 1, is seen to enhance the contribution of U_A to the overall uncertainty. Of course the opposite is also true; if the absolute value of the exponent were less than 1, the contribution of U_A would be lessened.

The above method for estimating U_r does not explicitly account for the actual probability distributions of the possible values of the parameters with associated uncertainty. Thus the U_r estimates obtained in this fashion are often faulty. In the

special case that these parameter values are normally distributed, it can be shown rigorously that the above expression gives the correct value for U_r, at least for algorithms of the form $r = Kx_1^a x_2^b x_3^c ....$ in which K is a constant and $a, b, c, ...$ are also constants (positive or negative). For other probability distributions for the possible parameter values and for more complicated algorithms, the U_r estimates obtained from Equations 5 or 6 may or may not compare well with the true U_r. Numerical evaluation of an algorithm using properly distributed values of the uncertain input parameters and statistical analysis of the distribution of evaluated algorithm results together appear to be the only method available for obtaining good estimates for the combined uncertainty U_r associated with the algorithm output. This is the approach used in BETAFACT. The actual implementation of the approach is described in Section 3.

Numerical evaluation of an algorithm allows one to model its associated uncertainty sources using essentially any distribution type desired. We have found it useful to model parameter and algorithm uncertainties with normal, lognormal, Beta, and uniform probability distributions. The forms and specific characteristics of these distribution types are summarized in the following subsection.

2.2 PROBABILITY DISTRIBUTIONS.

The distribution of possible values of each parameter or algorithm with associated uncertainty is modelled with a normal, lognormal, Beta, or uniform probability distribution in BETAFACT. Each of these probability distribution types has a more or less convenient mathematical representation. Before presenting these functions and summarizing some of their important characteristics, we very briefly review for the reader the meaning of a probability distribution and describe several measures generally used to characterize a probability distribution. We will see in Section 3 how the latter are related to a given specification of uncertainty for a parameter or algorithm.

Let $f(x)$ represent the probability distribution associated with the distributed variable x. Then $f(x)dx$ is the probability that x has a value in the interval between x and $x+dx$. Suppose we know variable x only takes on values in the range $a \leq x \leq b$. Since x must fall in this range, the integral of $f(x)$ over the full range of possible values is unity, provided $f(x)$ is normalized (which is the case for all four distributions we will consider below). Thus

$$\int_a^b f(x)dx = 1 \tag{9}$$

For some distributions, such as the uniform distribution, the limits a and b on the range of x variable values are bounded; for others, such as the normal distribution, they are not.

The mean μ (or expectation value $E(x)$) of a probability distribution is defined by the integral

$$\mu = E(x) = \int_a^b xf(x)dx \tag{10}$$

and the variance, $E(x-\mu)^2$, by the integral

$$E(x-\mu)^2 = \int_a^b (x-\mu)^2 f(x)dx \tag{11}$$

From the variance we obtain the standard deviation σ

$$\sigma = \sqrt{E(x-\mu)^2} \tag{12}$$

Now suppose we have collected a set of N values for the variable x. For instance, if x is a physical parameter, the set of values could be the measured results for x as found in many experiments. In the BETAFACT context, the set of values of interest is the set of results obtained in the evaluation of a algorithm many times. Let us further suppose that the set of x values have been sorted in ascending order (smallest to largest), the range from the smallest to largest value has been divided into M subintervals, and the number of times a value of the variable occurs in each subinterval has been determined. Let x_i be the variable value corresponding to the midpoint of the i-th subinterval and f_i be the number (or frequency of occurrence) of x values in the i-th subinterval. Clearly $N = \sum_{i=1}^{M} f_i$. Also if we plot f_i as a function of x_i, we obtain a graphical representation of the probability distribution for the variable x. Such a representation is often termed a histogram. If we have collected a sufficient number of data points and have used a reasonable number of subintervals in segregating the data into frequency of occurrence bins, then we might recognize the plotted distribution as being representable by one of the standard mathematical probability distributions.

The availability of the M pairs (x_i, f_i) enables us to determine a mathematical distribution which more or less closely replicates the actual distribution of variable values once we have selected a type of probability distribution to model the variable distribution. First we estimate the mean $\bar{x}$ of the variable distribution

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{M} x_i f_i \tag{13}$$

and the variance s^2

$$s^2 = \frac{1}{N}\sum_{i=1}^{M} (x_i - \overline{x})^2 f_i \tag{14}$$

The evaluated values $\tilde{x}$ and s are then used to fit the appropriate mathematical probability distribution to the distribution of x variable results using the relations between the mean and standard deviation and the corresponding distribution given in the following subsections for each distribution type in BETAFACT. We now turn to a brief description of the general form and characteristics of these distributions.

2.2.1 Normal Distribution.

The functional form of the normal or Gaussian probability distribution is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{15}$$

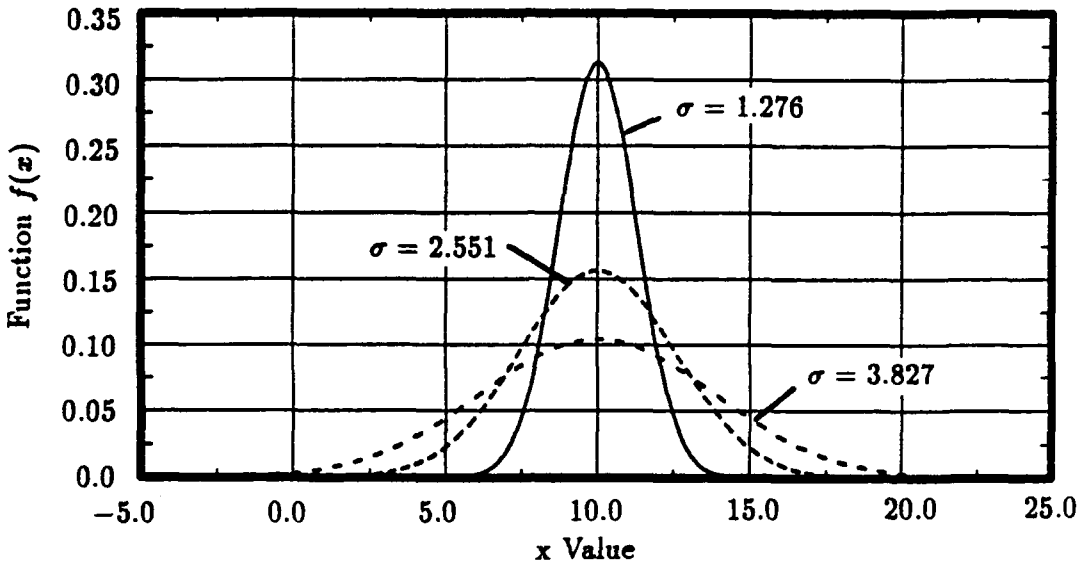

Figure 1. Example normal probability distributions with constant mean and variable standard deviation.

The mean of this distribution is μ, the variance is σ^2, and the standard deviation is σ. The range of possible variable values is unbounded ($-\infty < x < \infty$). The distribution is centered on and symmetric about $x = \mu$. If a variable x is normally distributed with mean μ and standard deviation σ, a randomly selected value for x will fall 95% of the time in the range ($\mu - 1.96\sigma \leq x \leq \mu + 1.96\sigma$). Only 2.5% of the time will the randomly selected value for x be below this range and 2.5% of the time above this range. Finally if we compute $\bar{x}(\simeq \mu)$ from Equation 13 and $x(\simeq \sigma)$ from Equation 14 for a set of variable values which appear to be normally distributed and then substitute these values into Equation 15, we obtain the normal distribution which will approximately fit the distribution of variable values. As an aid in visualizing the normal distribution and the effect on the distribution as σ is varied (with the mean fixed), Figure 1 shows 3 normal distributions for 3 different values of σ and a constant value for μ.

2.2.2 Lognormal Distribution.

Suppose variable x has a probability distribution of possible values which has a mean M. Further suppose the distribution for x is characterized by the existence of a positive constant $K(> 1.0)$ which has the property such that 95% of the time a randomly selected value for the variable x falls in the range $M/K \leq x \leq K \times M$ and equally as often above M as below M. Then the distribution of possible values for x is lognormally distributed if the distribution obtained by the change of variable $y = \ln x$ is a normal distribution in y with mean $\mu = \ln M$ and standard deviation $\sigma = \ln K/1.96$. In the BETAFACT (and FAST) context, K is recognized as the

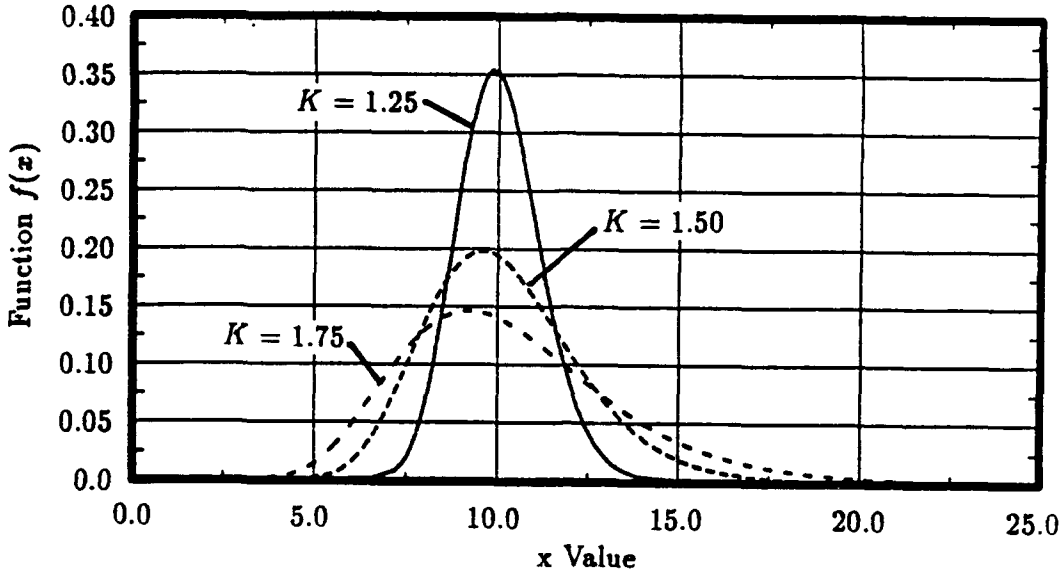

Figure 2. Example lognormal probability distributions with constant mean and variable standard deviation.

K-factor which characterizes (or quantifies) the uncertainty associated with variable x.

The typical form of the lognormal distribution is exhibited by the three example distributions shown in Figure 2. We see from this figure that the effect of increasing σ or K on the lognormal distribution is to shift the distribution peak to lower variable values and significantly stretch out the high end tail of the distribution.

2.2.3 Beta Distribution.

The Beta probability distribution is defined on the unit interval $0 \leq x \leq 1$ by the two parameter function

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \tag{16}$$

in which the normalization factor $B(\alpha,\beta)$ is given by

$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \tag{17}$$

In Equation 16, α and β are two parameters whose values influence the shape of the corresponding probability distribution. Both α and β are restricted to be greater than zero, and in fact in our experience with BETAFACT, all the Beta distributions we have used have had both α and β great than 1.

Figure 3 shows the Beta distributions obtained using 3 sets of α,β pairs. From the figure we see that if $\alpha = \beta$, the resulting probability distribution is symmetric and

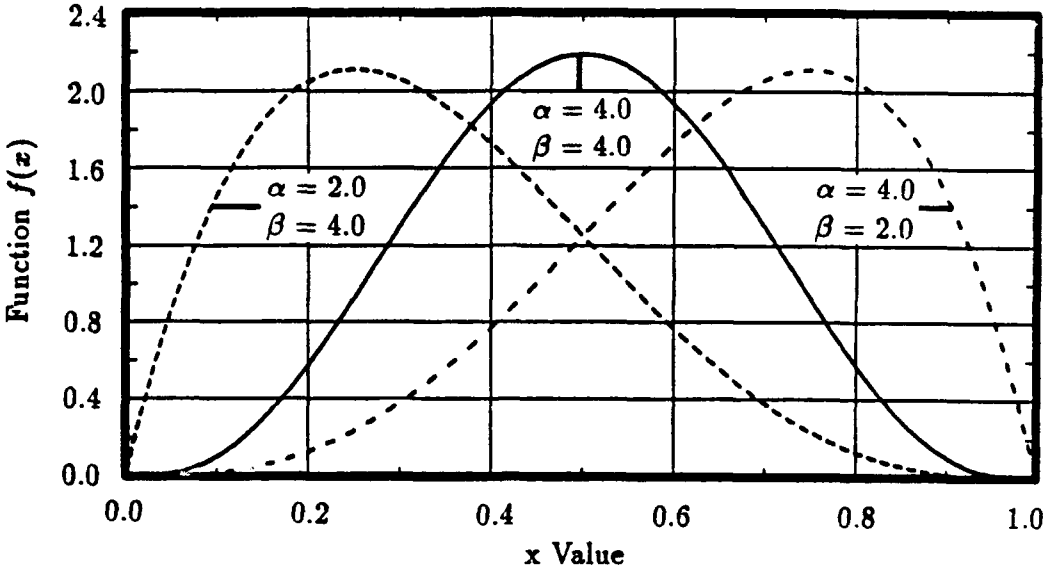

Figure 3. Example Beta probability distributions for different (α, β) parameter values.

looks somewhat like a normal distribution. If $\alpha < \beta$ (and $\alpha > 1.0$), the distribution is skewed to the left and the opposite is true if $\alpha > \beta$ (and $\beta > 1.0$). In the former case, the distribution has an appearance similar to a lognormal distribution. The ability to skew the Beta probability distribution to the right by proper choice of α and β is a feature of the function not shared by either the normal or lognormal distribution.

The mean $m(= \mu)$ of the Beta distribution with parameter values α and β is

$$m = \frac{\alpha}{\alpha + \beta} \tag{18}$$

and the variance is

$$\sigma^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{19}$$

Equations 18 and 19 can be inverted to give

$$\alpha = m\left(\frac{m - m^2}{\sigma^2} - 1\right) \quad \beta = (1 - m)\left(\frac{m - m^2}{\sigma^2} - 1\right) \tag{20}$$

However, since the standard deviation of the distribution must be positive, the following condition must also be satisfied (Simons, D.A., 1988):

$$\sigma \leq min\left[m\left(\frac{1 - m}{1 + m}\right)^{1/2}, (1 - m)\left(\frac{m}{2 - m}\right)^{1/2}\right] \tag{21}$$

The implications of the above restriction on σ will be considered in the section of this report in which implementation of Beta distributions in BETAFACT is discussed.

If we have a set of variable results which we think can be represented with a Beta distribution, Equations 16 and 20 indicate how this might be done. First we evaluate the mean m and the variance σ^2 of the set of variable results and use Equation 20 to evaluate corresponding values for α and β. Then we evaluate Equation 16 to obtain the desired distribution. This is one approach used in BETAFACT. A somewhat less demanding (and less accurate) method is also implemented in the code. This method will be described in Section 3.

2.2.4 Uniform Distribution.

A generalized form of the uniform probability distribution is used in BETAFACT. The function describing this distribution is defined on the interval $a \leq x \leq b$ as

$$f(x) = \begin{cases} \frac{1}{2}\frac{1}{m-a} & a \leq x \leq m \\ \frac{1}{2}\frac{1}{b-m} & m \leq x \leq b \end{cases} \tag{22}$$

The parameter m corresponds to the value of x such that 50% of the time the value of x is less than m. The mean μ of the distribution is

$$\mu = \frac{a+b+2m}{4} \tag{23}$$

and the variance σ^2 is

$$\sigma^2 = \frac{1}{6}\left[a^2+b^2+m(a+b)+2m^2\right] - \frac{1}{16}\left[a+b+2m\right]^2 \tag{24}$$

For $m = \frac{1}{2}(a+b)$ these reduce to

$$\mu = \frac{1}{2}(a+b) \quad \sigma^2 = \frac{1}{12}(b-a)^2 \tag{25}$$

as required for a symmetric uniform distribution. Figure 4 shows examples of the generalized uniform probability distribution associated with three different values of m for fixed a and b. Clearly, given a set of variable values which can be modelled with a generalized uniform distribution, we need to determine only a, b, and m for the set of variables to fit the required uniform distribution.

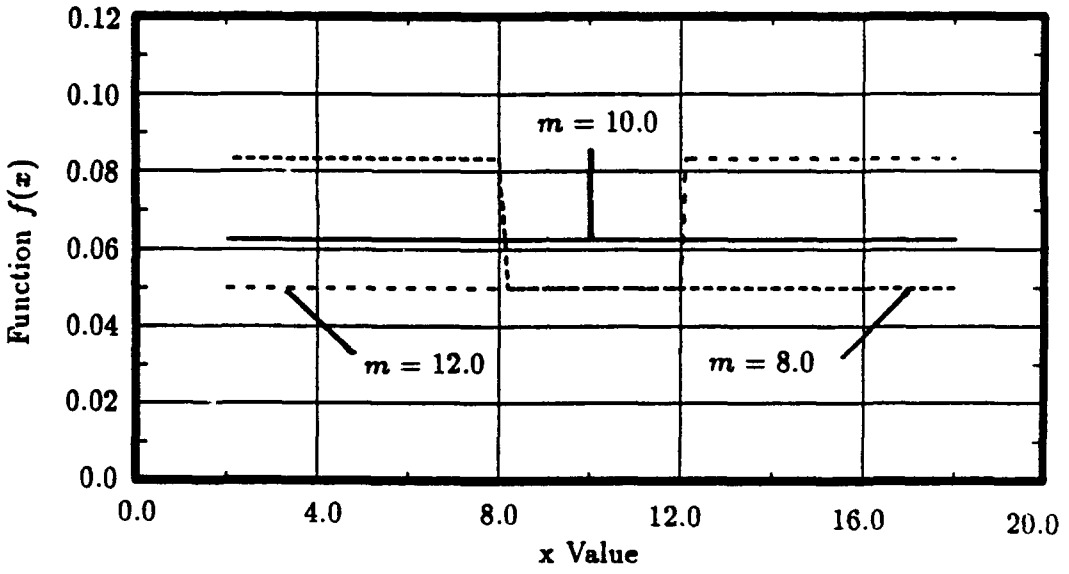

Figure 4. Example generalized uniform probability distributions.

SECTION 3
IMPLEMENTATION

Four types of probability distributions (normal, lognormal, Beta, and uniform) are available in BETAFACT for modelling the distributions of parameters and algorithms which have associated uncertainty. Some important characteristics of each of these distributions are summarized in Section 2 of this report. The purpose of this section is to describe the specifics of how the code uses the probability distributions to model uncertainties, and the mechanics involved in developing the algorithm results distribution which ultimately provides the information needed to quantify the overall uncertainty associated with an algorithm output.

Several steps are involved in the processing of an algorithm with BETAFACT. These steps, with minor modifications, need to be performed for any algorithm analyzed with the code. The steps are briefly described in subsection 3.1. The precise manner in which some of the steps are implemented in BETAFACT is dependent on the probability distribution type used to model the uncertainties. Details of the implementation are described, in turn, for each of the 4 probability distribution types in subsections 3.2 through 3.5.

3.1 ANALYSIS PROCESS OVERVIEW.

In the following we assume that the algorithm to be analyzed with BETAFACT is reduced to the form of Equation 1; i.e. the algorithm output is expressed strictly in terms of input parameters, any number of which have associated uncertainty. It has been our experience that at least one parameter in the algorithm may be regarded as an independent parameter which does not have associated uncertainty. BETAFACT expects there to be at least one such parameter. If there isn't explicitly such a parameter, one can easily be introduced as a multiplier, with value of unity, which acts on the entire right hand side of Equation 1.

To be available to BETAFACT, the algorithm must be coded in FORTRAN in a subroutine (named TF) called by the code. This subroutine must be compiled and linked with the BETAFACT object file to obtain the algorithm specific executable form of the program. Details of how this is accomplished are given in Section 5 of this report.

We have found on occasion that some parameters, either with or without uncertainty, which are used in an algorithm are most conveniently obtained from subordinate algorithms. For instance, the thickness parameter h in the algorithm of Equation 3 may be a known function of the impulse intensity I since the impulsive loading may cause the cylinder wall to experience back-surface spall. Thus in this case, to obtain the correct value of h for use in Equation 3, we would first need to evaluate the functional (or algorithmic) relationship between h and I. BETAFACT provides for this situation by allowing the user to write another FORTRAN subroutine (called

AUXALGO) which is called by the program. This subroutine, too, must be compiled and linked to BETAFACT, in the manner described in Section 5, to become available to the code. Specific details concerning the structure and use of both the TF and AUXALGO subroutines are provided in Section 5.

Once the executable file of BETAFACT containing the user's algorithm is prepared, analysis of the algorithm by the code proceeds in an interactive mode. The program prompts the user for all additional information required to initially analyze the algorithm or re-analyze it according to user specified modifications. The additional information required include identification of the probability distribution type to be used in modeling parameter and algorithm uncertainties, the nominal value for each parameter with associated uncertainty and the specification (in terms of K-factors) of that uncertainty, the values of independent parameters, and the number of times the algorithm is to be evaluated to develop the results distribution.

After the problem to be processed is completely defined, BETAFACT first uses the input data to determine the precise probability distribution for each parameter and algorithm with uncertainty which will be used to model the uncertainty. The code then proceeds to evaluate the algorithm the requested number of times. For each evaluation, a different normally or uniformly (as appropriate) distributed random number is used in conjunction with a parameter nominal value and associated probability distribution to obtain a randomized value for each parameter with uncertainty. Details of this process are described in the subsections below. The algorithm is then evaluated with the set of randomized parameters, uncertainty is applied to the result of the computation (if required), and the final result is then saved. In this manner the algorithm results distribution is generated.

The algorithm results distribution contains all the information required for determining the combined uncertainty associated with the output of the algorithm. First the results distribution is sorted and ordered from lowest to highest value. Next its mean, variance, and standard deviation are evaluated using Equations 13 and 14. These quantities (and in some cases the low and high extreme values of the algorithm results distribution) are all the information needed to quantify the K- factor uncertainty associated with the results distribution. These K-factors are reported by the program and then the user is prompted to specify whether or not data files containing the histogram of the results distribution and a probability distribution fit to the histogram are to be generated. These files contain data pairs defining the histogram and the fit and can be plotted with software external to BETAFACT. The code finally prompts the user to interactively modify the problem and re-analyze it or to terminate the program execution.

The above description of the algorithm analysis process used in BETAFACT emphasizes that input uncertainties and the overall algorithm output uncertainty are in terms of what are called K-factors. We first briefly mentioned the K-factor approach for quantifying uncertainty in our description of the lognormal distribution in subsection 2.2.2. In the following subsections, we will see that the meaning of the K-factor specification for an uncertainty is dependent on the probability distribution used to model that uncertainty. It is crucial that the user of BETAFACT thoroughly

understand the K-factor uncertainty concept in order to be able to properly apply the code and interpret its results.

3.2 NORMALLY DISTRIBUTED UNCERTAINTY.

Suppose that a parameter x with associated uncertainty is known to be or is modelled as being normally distributed. If we assume the mean μ and standard deviation σ of this parameter distribution are known, then the probability distribution for the parameter is given by Equation 15. We recall that for a randomly selected value of x, 95% of the time the selected value will fall in the range $\mu - 1.96\sigma \leq x \leq \mu + 1.96\sigma$ and equally as often above μ as below μ. The definition of the K-factor specification of the uncertainty associated with this parameter is

$$K = 1.0 + 1.96\frac{\sigma}{\mu} \tag{26}$$

To use parameter x in BETAFACT, we specify it as being normally distributed and enter its nominal value $ANOM = \mu$ and K-factor uncertainty $AK = K$. (Here and in the following we will use when possible variable names and expressions similar or identical to those actually coded in BETAFACT.) Then the low and high side values of x corresponding to the 95% range given above in terms of μ and σ are, respectively, $ALO = 2 * ANOM - AHI$ and $AHI = AK * ANOM$. Finally, if RN is a normally distributed random number, then a randomized value A for parameter x is given by

$$A = ANOM + RN * (AHI - ANOM)/1.96 \tag{27}$$

Once an algorithm results distribution is generated using normally distributed parameter and/or algorithm uncertainties, the mean μ and standard deviation σ of the distribution is computed. Equation 26 is then used to estimate the appropriate K-factor uncertainty which should be associated with the output of the algorithm.

3.3 LOGNORMALLY DISTRIBUTED UNCERTAINTY.

Suppose parameter x is lognormally distributed with mean $ANOM$. We recall from subsection 2.2.2 that if 95% of the time a randomly selected value for x falls in the range $ANOM/K \leq x \leq K \times ANOM$, and equally as often below $ANOM$ as above it, then $AK = K$ is the K-factor uncertainty specification for x. The lognormal distribution for x can be transformed into a normal distribution in $y = \ln x$ with mean and standard deviation given by

$$\mu = \ln ANOM \qquad \sigma = \frac{\ln K}{1.96} \tag{28}$$

The low and high side values, respectively, of x corresponding to the 95% range given above are clearly $ALO = ANOM/K$ and $AHI = K * ANOM$. A randomized

value A for parameter x is given by

$$A = ANOM * ALO * *(RN/1.96) \tag{29}$$

in which RN is a normally distributed random number.

After we compute the mean μ and standard deviation σ of an algorithm results distribution, the K-factor estimate to be associated with the algorithm output is given by $K = \exp(1.96\sigma/\mu)$

3.4 BETA DISTRIBUTED UNCERTAINTY.

Parameter x is Beta distributed if its distribution of possible values is given by Equation 16. This distribution is not as conveniently used as the normal and lognormal distributions. In fact, we don't use the Beta probability distribution as such in BETAFACT, but instead we use the cumulative probability distribution (CPD) associated with a given Beta distribution. The CPD is simply the total area (cumulative probability) under the normalized Beta distribution curve, integrated from $y = 0$ to $y = y'$, as a function of y'. We further use a discrete form of the CPD associated with each Beta distribution which is referenced in the course of modeling uncertainties. The discrete CPD in each case consists of a list of the 21 values for y' corresponding, respectively, to 0%, 5%, 10%, ..., 100% of the cumulative probability. The eleventh entry in such a list corresponds to the fraction of the unit interval (0 to 1) at which 50% of the area under the Beta probability distribution curve is to the left and 50% is to the right of that unit interval location.

We complicate matters further by offering two options in BETAFACT regarding the modelling of parameter uncertainties with Beta CPDs; the option to use the most appropriate Beta CPD from a library of tabulated distributions or to have the code calculate a Beta CPD specifically for the situation at hand. The former option is the only one currently offered in the FAST code. The second option increases the accuracy with which uncertainties can be modelled using Beta CPDs. As will be seen below, however, there is a cost associated with each option.

Before considering in greater detail each option, we note that both require the specification of two K-factors for each Beta distributed parameter. A K-factor is required to characterize each side of the distribution relative to the nominal value $ANOM$ of the parameter. The low side K-factor is named $AKLO$ in BETAFACT and the high-side K-factor $AKHI$. The meaning of these K-factors is that 100% of the possible values of parameter x fall in the range $ANOM/AKLO \leq x \leq AKHI * ANOM$, 50% of the time below $ANOM$ and 50% of the time above. Thus the low and high side extreme values of the parameter are $ALO = ANOM/AKLO$ and $AHI = AKHI \times ANOM$, respectively.

A randomized value A for the parameter is obtained in a somewhat complicated process using the 21 discrete y' values for a Beta CPD which are stored in an array called BD. First a uniformly distributed random number RN is generated and used to compute the number $FLI = 20.0 * RN + 1.0$. This latter number is decomposed

into its integer part (0,1,2,...,20) and its fractional part. The integer part of FLI is used to select 2 adjacent entries of the BD array. For instance, if the integer part of FLI is i, then the i-th and $(i+1)$ -th BD entries are selected. The fractional part F quantifies a fraction of the distance between BD(i) and BD($i+1$). It is used to calculate the value of variable BDV;

$$BDV = (1 - F) * BD(i) + F * BD(i + 1) \tag{30}$$

which is then used to evaluate the randomized value for the variable corresponding to the random number RN;

$$A = ALO + BDV * (AHI - ALO) \tag{31}$$

The easiest (and less accurate) method available in BETAFACT for modelling uncertainties with Beta probability distributions is to use the Beta CPDs which are tabulated in the code. There are 81 of these CPDs. Table 1 identifies the medians of the distributions and the values of the parameters α and β which were used to generate the distribution. The method we use in the program to associate a specific tabulated Beta CPD to a parameter with uncertainty is a two step process. First we compute the median (TEMP) on the unit interval defined by the nominal value $ANOM$ and K-factors ($AKLO$ and $AKHI$) of the uncertain parameter:

$$TEMP = \frac{ANOM - ANOM/AKLO}{AKHI * ANOM - ANOM/AKLO} = \frac{1 - 1/AKLO}{AKHI - 1/AKLO} \tag{32}$$

We then determine which of the 81 tabulated Beta CPDs has a median (11-th entry) closest to TEMP. The closest CPD thereby identified is used to model the uncertainty associated with the parameter.

The more accurate method in BETAFACT for modeling Beta distributed parameter uncertainty is to have the code compute the Beta CPD which corresponds exactly to the distribution desired. This is achieved by specifying for an uncertain parameter not only its nominal value $ANOM$ and associated K-factors $AKLO$ and $AKHI$, but also the standard deviation σ_q. The latter must satisfy the constraint given in Equation 21 which BETAFACT computes and reports to the user. With these four inputs, then, and using $ALO = ANOM/AKLO, AHI = AKHI \times ANOM$, values for m and σ are computed (Simons, 1988),

$$m = \frac{ANOM - ALO}{AHI - ALO} \qquad \sigma = \frac{\sigma_q}{AHI - ALO} \tag{33}$$

and then Equation 20 is evaluated to determine corresponding values for α and β. Finally Equation 16 (with Equation 17) is evaluated and the corresponding Beta CPD is computed and stored in the BD array used for modeling the parameter uncertainty.

BETAFACT generates three estimates for the K-factors to be associated with a Beta distributed algorithm results distribution with mean μ and low and high extremes XLO and XHI, respectively. These are the low-side K-factor estimate, $KLO = \mu/XLO$, the high-side K-factor, $KHI = XHI/\mu$, and the average of the two $KAVE = 0.5(KLO + KHI)$.

3.5 UNIFORMLY DISTRIBUTED UNCERTAINTY.

The generalized uniform probability distribution for parameter x is given by Equation 22. This distribution is characterized in terms of the nominal value $ANOM$ for the parameter and low and high side K-factors, $AKLO$ and $AKHI$, respectively. These K-factors mean that 100% of the possible range for parameter x is given by $(ANOM/AKLO \leq x \leq AKHI * ANOM)$ with 50% of the distribution below $ANOM$ and 50% above. As in the case of a Beta distributed parameter, the extreme values for the parameter are $ALO = ANOM/AKLO$ and $AHI = AKHI * ANOM$.

A uniformly distributed random number RN is used to obtain a randomized value A for parameter x. If the value of RN is less than or equal to 0.5, then

$$A = ALO + 2.0 * RN * (ANOM - ALO) \tag{34}$$

else

$$A = ANOM + 2.0 * (RN - 0.5) * (AHI - ANOM) \tag{35}$$

The same three K-factor estimates produced by BETAFACT for Beta distributed algorithm results are also generated for algorithm results distributions computed using uniform distributions to model parameter and algorithm uncertainties.

Table 1. Tabulated Beta cumulative probability distribution means and α and β parameters.

Dist. Numb.	Median	α	β	Dist. Numb.	Median	α	β
1	0.167	1.04	4.0	41	0.500	4.0	4.0
2	0.172	1.07	4.0	42	0.512	4.0	3.83
3	0.179	1.11	4.0	43	0.524	4.0	3.66
4	0.185	1.14	4.0	44	0.535	4.0	3.52
5	0.192	1.18	4.0	45	0.545	4.0	3.39
6	0.200	1.23	4.0	46	0.556	4.0	3.26
7	0.208	1.28	4.0	47	0.565	4.0	3.15
8	0.217	1.33	4.0	48	0.574	4.0	3.05
9	0.227	1.39	4.0	49	0.583	4.0	2.95
10	0.238	1.46	4.0	50	0.592	4.0	2.86
11	0.250	1.54	4.0	51	0.600	4.0	2.77
12	0.256	1.58	4.0	52	0.608	4.0	2.69
13	0.263	1.63	4.0	53	0.615	4.0	2.62
14	0.270	1.68	4.0	54	0.623	4.0	2.55
15	0.278	1.73	4.0	55	0.630	4.0	2.48
16	0.286	1.79	4.0	56	0.636	4.0	2.42
17	0.294	1.85	4.0	57	0.643	4.0	2.36
18	0.303	1.92	4.0	58	0.649	4.0	2.31
19	0.313	1.99	4.0	59	0.655	4.0	2.26
20	0.323	2.07	4.0	60	0.661	4.0	2.20
21	0.333	2.16	4.0	61	0.667	4.0	2.16
22	0.339	2.20	4.0	62	0.677	4.0	2.07
23	0.345	2.26	4.0	63	0.687	4.0	1.99
24	0.351	2.31	4.0	64	0.697	4.0	1.92
25	0.357	2.36	4.0	65	0.706	4.0	1.85
26	0.364	2.42	4.0	66	0.714	4.0	1.79
27	0.370	2.48	4.0	67	0.722	4.0	1.73
28	0.377	2.55	4.0	68	0.730	4.0	1.68
29	0.385	2.62	4.0	69	0.737	4.0	1.63
30	0.392	2.69	4.0	70	0.744	4.0	1.58
31	0.400	2.77	4.0	71	0.750	4.0	1.54
32	0.408	2.86	4.0	72	0.762	4.0	1.46
33	0.417	2.95	4.0	73	0.773	4.0	1.39
34	0.426	3.05	4.0	74	0.783	4.0	1.33
35	0.435	3.15	4.0	75	0.792	4.0	1.28
36	0.444	3.26	4.0	76	0.800	4.0	1.23
37	0.455	3.39	4.0	77	0.808	4.0	1.18
38	0.465	3.52	4.0	78	0.815	4.0	1.14
39	0.476	3.66	4.0	79	0.821	4.0	1.11
40	0.488	3.83	4.0	80	0.828	4.0	1.07
				81	0.833	4.0	1.04

SECTION 4
REPRESENTATIVE ANALYSES

Besides having been used extensively in our lethality assessment activities for the LTH-3 Program, BETAFACT has been exercised on many simple problems to verify the implementation in the code of the probability distributions described in Section 2 and the methodology of Section 3. It has also been used to analyze more complicated problems which more completely illustrate the several uses of the code. The results of several simple BETAFACT analyses are reviewed in subsection 4.1. The application of the code in the analysis of a more realistic algorithm is the central theme of subsection 4.2.

4.1 VERIFICATION ANALYSES.

We consider the simplest possible algorithm $r = x$ in analyses to verify proper implementation of the theory and methodology in BETAFACT. This algorithm has the output r and the single input parameter x. The input parameter is assumed to have associated uncertainty. Towards the end of this subsection, we also consider the case in which the algorithm has associated uncertainty.

A listing of subroutine TF used to input this simple algorithm to BETAFACT is given in Appendix A. The subroutine locally references variable x (which is passed to the subroutine as the first element $A(1)$ in array A as $ADUM$. The subroutine also uses the independent parameter $XVAL(1)$ which has an interactively specified value of 1.0 and hence has no effect on the algorithm output $r = YVAL$. More details about formulating subroutine TF are given in subsection 4.2.

In the situation when uncertainty is associated only with the input parameter x, we know that the distribution of the algorithm results should be identical to that used to model the input parameter uncertainty. In the following the latter is modelled, in turn, using each of the four probability distribution types available in BETAFACT. We assume a nominal value for the parameter of 50,000 and analyze the problem for a few different values of the K-factor uncertainty. The algorithm is evaluated 5000 times in each of the referenced analyses in order to generate the results distribution. A fewer number of algorithm evaluations (e.g., 1000) gives results not substantially different from those we present below.

The initial set of analysis results we consider are obtained when the parameter uncertainty is modelled as normally distributed with a K-factor specification of 1.25 or 1.50. Plots of the computed (solid line) and fitted (dash line) results distribution for the two K-factor specifications are shown in Figure 5. The figures also report the random number generator seed integer used at the start of each analysis. The value of this seed integer is purposely and arbitrarily varied from one analysis to another.

The plots in Figure 5 show that the computed results distribution and the fitted distribution generally have similar shapes. No significant effort is made in BETAFACT

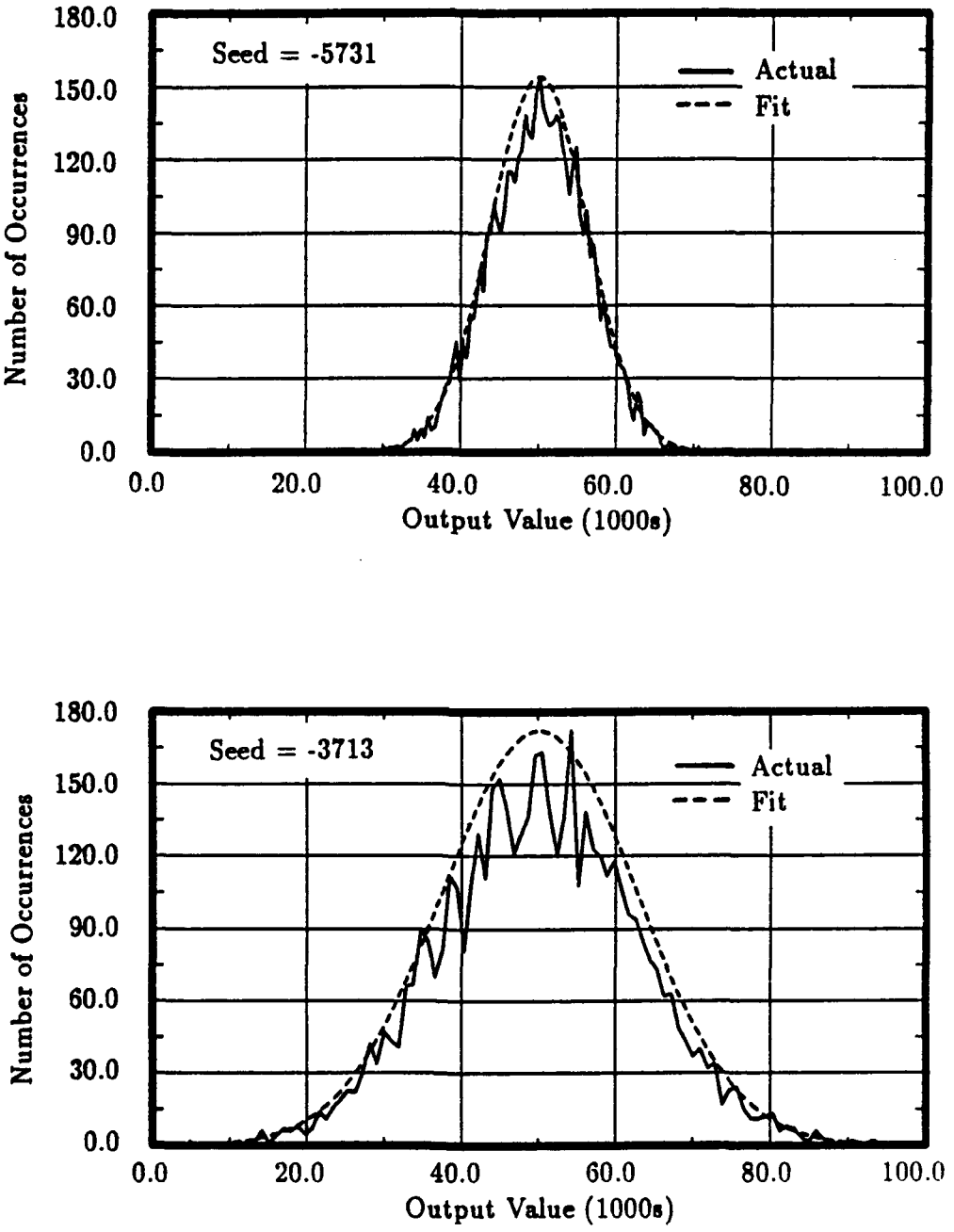

Figure 5. Simple algorithm results distributions for $K = 1.25$ (top) and $K = 1.50$ (bottom) and normally distributed uncertainty.

to obtain the "best fit" to the computed results distribution; a simple scaling procedure is employed to approximately match the peak of the distribution. The purpose of the plotted distributions is more of a qualitative rather than a quantitative check of the results. What we are really after in using BETAFACT is the K-factor associated with the results distribution and the mean of the distribution. These two, together with the probability distribution type, are sufficient to completely characterize the uncertainty distribution associated with the algorithm output for subsequent use in FAST. Thus to verify that the code is functioning properly, we must compare the computed statistics of the results distribution to the corresponding theoretical values.

In defining this simple problem, we specified the nominal value of the parameter as $\mu = 50,000$ and its K-factor uncertainty as 1.25 or 1.50. Equation 26 enables us to relate these values to the corresponding standard deviation σ of the normal probability distribution which exactly models the parameter and hence the algorithm results uncertainty. Doing so, we find that the exact values for σ are 6378 for $K = 1.25$ and 12,755 for $K = 1.50$. The values actually computed by BETAFACT for μ and σ and for the K-factor estimate of the results distribution are reported in Table 2. Compared to the theoretical values, the computed results are seen to be quite good. Two reasons the comparisons aren't better are that an infinite number of algorithm evaluations were not used to generate the results distribution (a small effect here) and only 100 distinct frequency of occurrence bins were used to construct the histograms (most significant effect).

The above simple analysis problem was repeated, this time modelling the parameter uncertainty distribution as lognormal. The plotted results from these analyses are shown in Figure 6. The equation at the end of subsection 3.3 enables us to relate the specified K-factor and mean to the corresponding theoretical standard deviation. The theoretical values are compared to the BETAFACT computed values in Table 2. The latter are within a few percent of the theoretical values. The reasons given above for the normal distribution computed results discrepancy apply here as well.

We next analyzed the simple algorithm using Beta cumulative probability distributions to model the parameter uncertainty. The two analyses described above were each performed twice, first using tabulated Beta CPDs and then using BETAFACT computed distributions. The reader may recall that a standard deviation for the uncertain parameter must be specified to allow the use of computed rather than tabulated distributions. The standard deviations computed when the uncertainties were modelled with tabulated distributions were used as the exact input standard deviations for the calculated distribution analyses. The computed results for both sets of analyses are presented in Table 2. Figure 7 shows the distribution plots obtained in the two computed Beta distribution analyses. The tabulated distribution results are nearly identical to those shown in the figure.

Finally, we analyzed the simple algorithm example using a uniformly distributed uncertainty model. Three combinations of K-factors were considered. For the first two K-factor combinations ((1.25, 1.20) and (2.00, 1.50)) Equation 25 provides the theoretical value of the corresponding standard deviation. Equation 24 must be used

Table 2. Simple algorithm analysis results.

Distribution Type	Specified K-Factors		Exact	Computed			
	KLO	KHI	σ	$\mu^{\ddagger}$	σ	KLO	KHI
Normal	1.25	-	6378	49976	6249	1.245	-
	1.50	-	12755	50152	12769	1.499	-
Lognormal	1.25	-	5692	50316	5829	1.255	-
	1.50	-	10344	51167	10674	1.505	-
Beta†							
-Tabulated	1.25	1.25	-	50169	4223	1.254	1.245
	1.50	1.50	-	50434	7877	1.513	1.487
-Calculated	1.25	1.25	4223	50054	4489	1.251	1.249
	1.50	1.50	7877	50111	8284	1.503	1.496
Uniform	1.25	1.20	5774	49998	5806	1.250	1.200
	2.00	1.50	14434	50002	14500	1.999	1.500
	2.00	2.00	21949	48995	23046	1.960	2 .041

†- Analyses performed first with tabulated (unspecified σ) and then with calculated (specified σ) Beta cumulative probability distributions.

‡- Theoretical mean is 50000 for all cases.

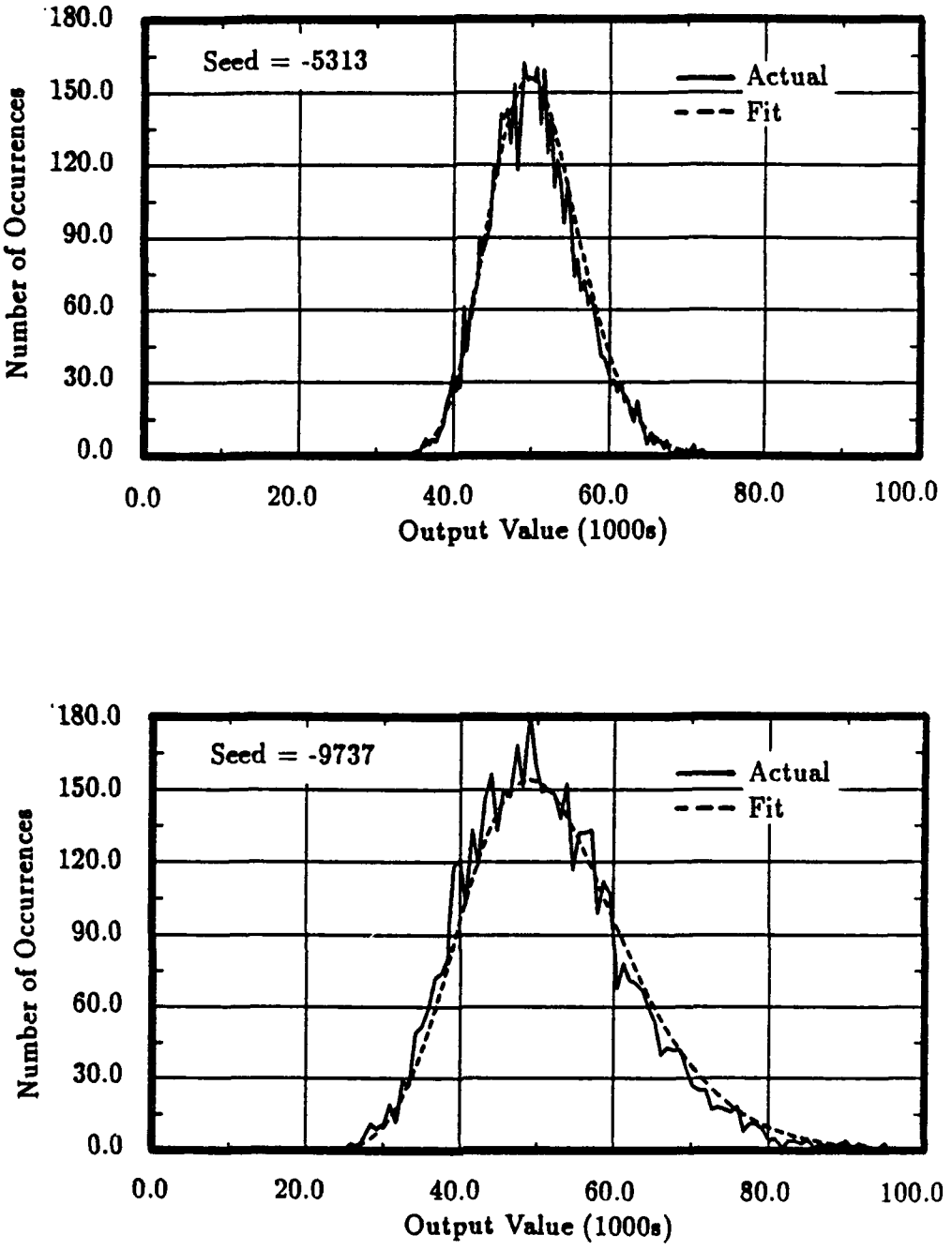

Figure 6. Simple algorithm results distributions for $K = 1.25$ (top) and $K = 1.50$ (bottom) and lognormally distributed uncertainty.

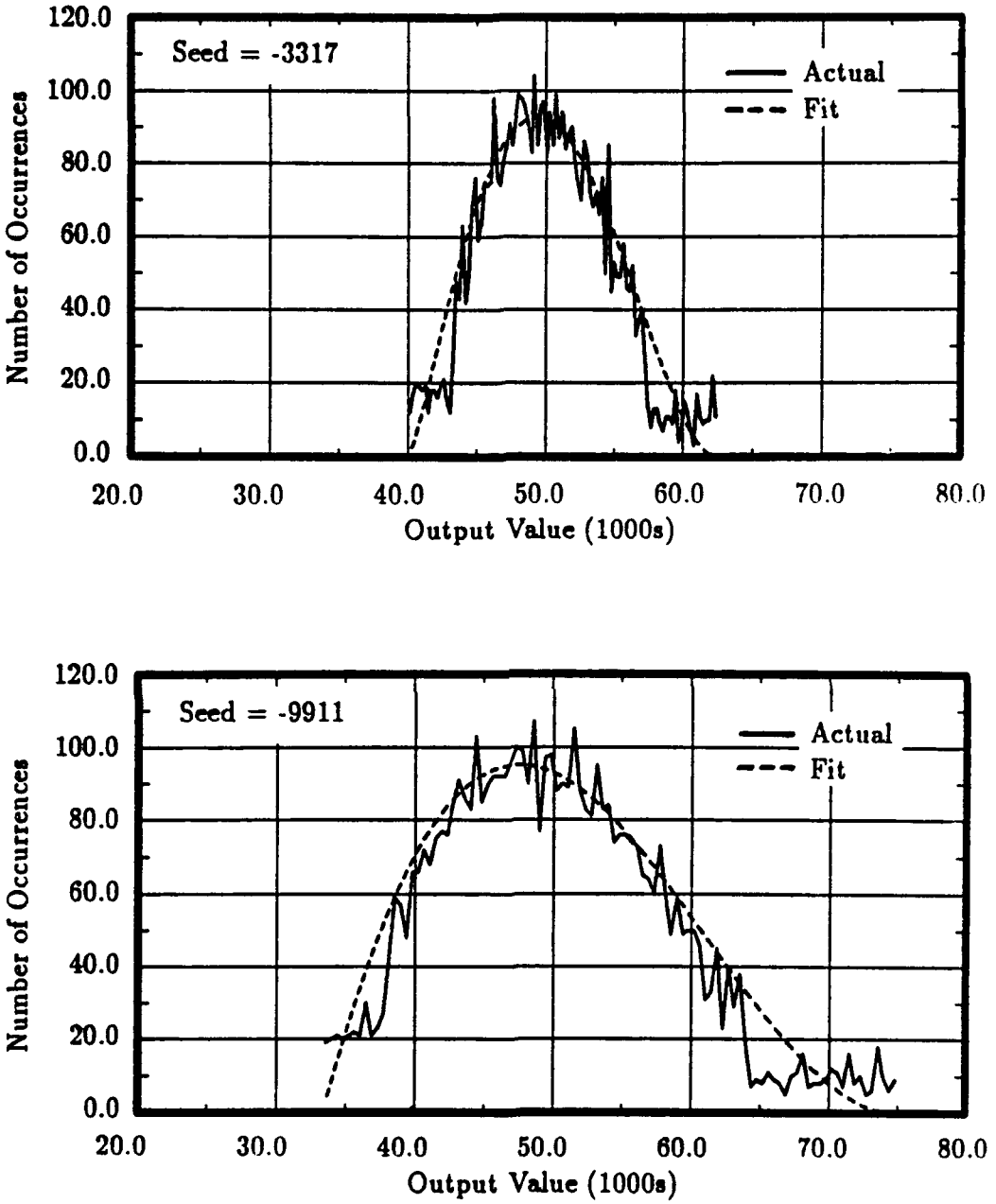

Figure 7. Simple algorithm results distributions for $K = 1.25$ (top) and $K = 1.50$ (bottom) and Beta distributed uncertainty.

to evaluate the theoretical standard deviation associated with the third K-factor pair (2.00, 2.00). The theoretical and BETAFACT computed values for the mean, standard deviation, and K-factors of these analyses are given in Table 2. The agreement between computed and theoretical values is quite good. Figures 8 and 9 show plots of the computed and fitted results distributions.

While the plots in Figures 5 through 9 are useful in visualizing the output from BETAFACT, it is the comparisons evident in Table 2 which indicate that the code is functioning properly in the analyses of the simple algorithm $r = x$. These comparisons show that the code adequately replicates the K-factors and nominal value specified for the uncertain algorithm parameter. It is somewhat less successful in reproducing the theoretical standard deviation. The limited structure (100 bins) of the histograms compiled from the computed algorithm results distribution is believed to be the main source for the error in the computed standard deviations.

As a final verification analysis, we considered the case in which both the input parameter and the algorithm have associated uncertainty. If both of these uncertainty sources are modeled as normally distributed, then the theoretical result is that the variance (standard deviation squared) of the results distribution is equal to the sum of the variance associated with the input parameter and that associated with the algorithm. Specifying the parameter nominal value again as 50,000, the parameter K-factor as 1.25, and the algorithm K-factor also as 1.25, the parameter standard deviation is 6378 (as is that of the algorithm), and the results distribution has a theoretical standard deviation of 9020 (and corresponding K-factor of 1.354). When this problem is analyzed with BETAFACT using 5000 evaluations, we obtain the following computed values; $\mu = 50,067, \sigma = 9088$, and $K = 1.356$. The computed values are in very good agreement with the theoretical results. Plots of the computed histograms for this case are shown in Figure 10.

4.2 REALISTIC ALGORITHM EXAMPLE.

The analysis results presented in the previous subsection give us confidence that the probabilistic theory we are using is correctly implemented in BETAFACT. In the current subsection we illustrate how the code is used to analyze a realistic algorithm. Specifically we consider a slightly modified version of the algorithm presented previously in Equation 3. The modified version of this basic algorithm is as follows:

$$\varepsilon = \frac{I_r^2}{2\rho\sigma h_r^{1.5}} \tag{36}$$

This algorithm provides an estimate for the peak strain ε in a metallic cylinder in terms of the parameters on the right hand side of the equation. These parameters are material density ρ, material ultimate strength σ, thickness h_r and peak impulse intensity I_r. We assume that each parameter used in the right hand side of the algorithm, as well as the algorithm itself, has associated uncertainty.

The algorithm given in Equation 36 is different from the Equation 3 algorithm in three respects. First, we have arbitrarily changed the exponent of the thickness

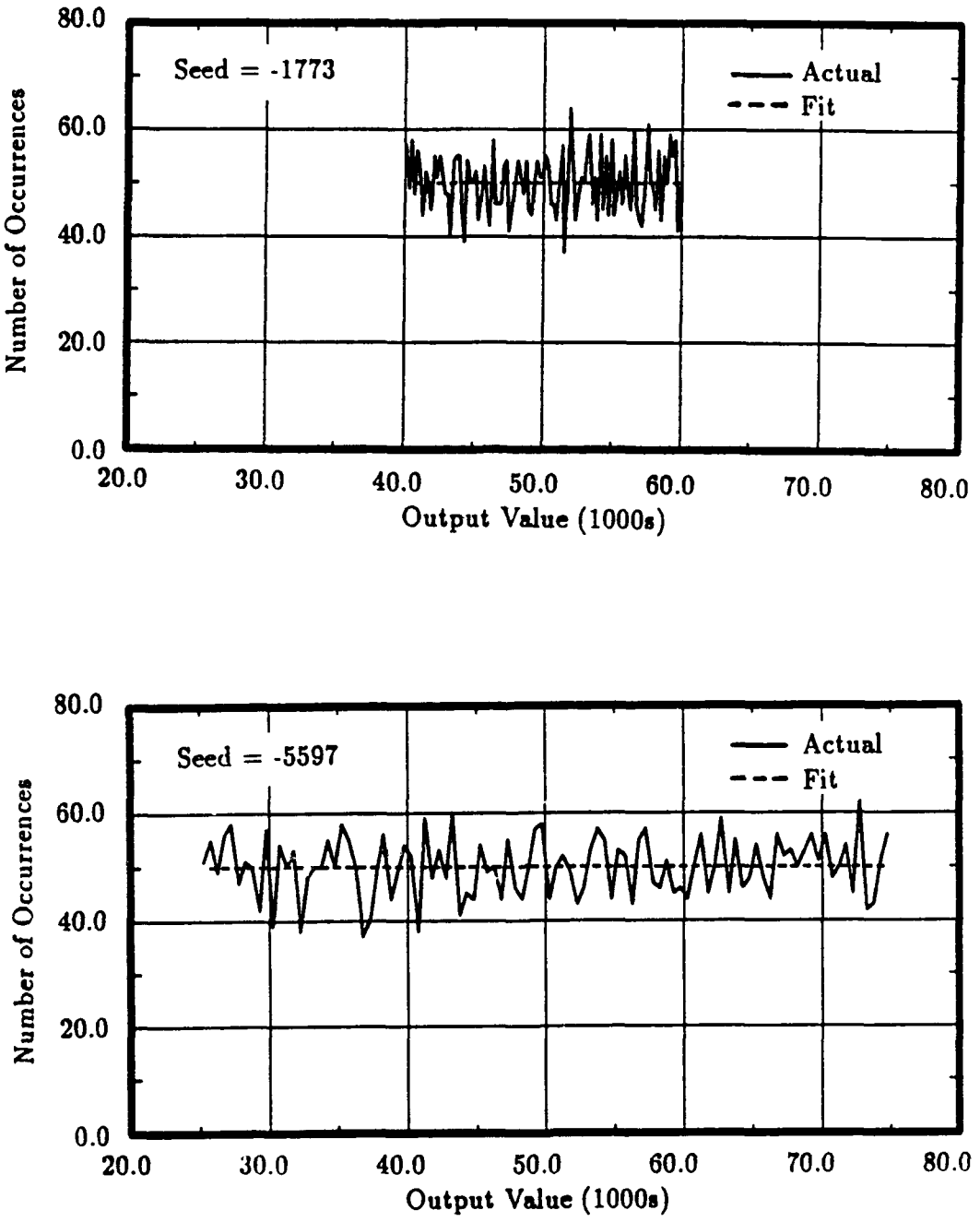

Figure 8. Simple algorithm results distributions for K-factor pairs (1.25, 1.20) (top) and (2.00, 1.50) (bottom) and uniformly distributed uncertainty.

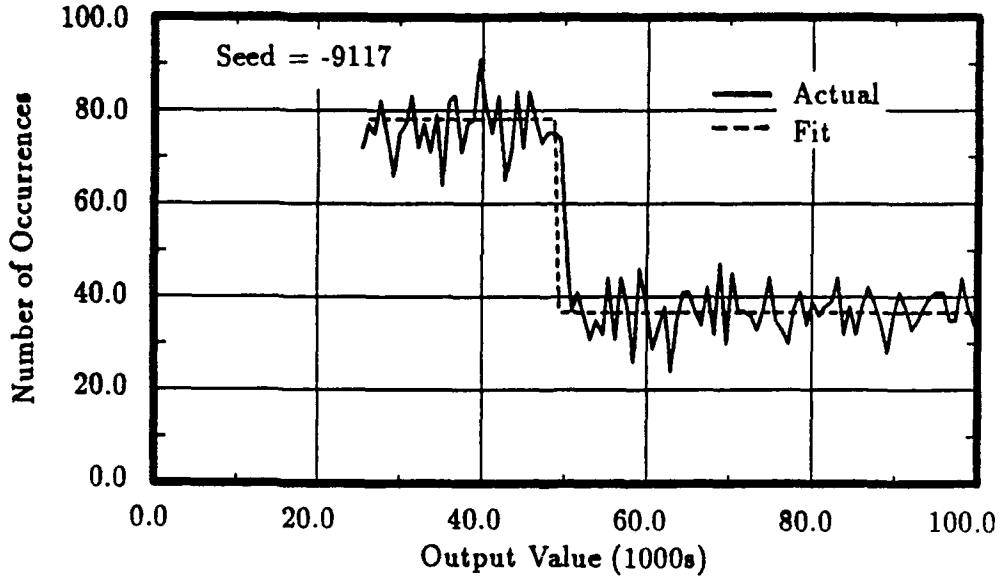

Figure 9. Simple algorithm results distributions for K-factor pair (2.00, 2.00) and uniformly distributed uncertainty.

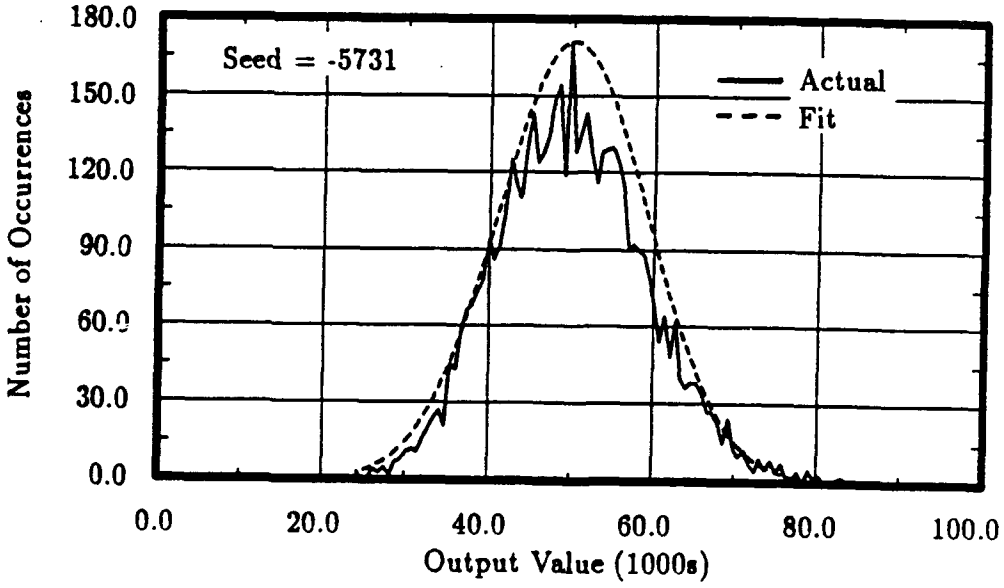

Figure 10. Simple algorithm results with combined normally distributed uncertainties.

parameter from 2 to 1.5. We have then added the subscript r to h and I to indicate that the actual thickness of the cylinder wall and the actual impulse intensity which loads the structure are not used in the algorithm. Instead we assume we must first compute these parameter values from auxiliary algorithms before the proper values can be used in the strain algorithm. In this example analysis, we assume the following forms for these auxiliary algorithms;

$$I_r = [3.5 + 0.08(I - 3.5)]\,(2h)^{1/2} \tag{37}$$

and

$$h_r = h - [0.05 + 0.001(I - 3.5)]\,(2h)^{1/2} \tag{38}$$

In these equations h is the actual thickness of the cylinder wall before the impulsive load is applied and I is the peak intensity of the actual applied impulsive load. The two auxiliary algorithms are used here to model the situation in which the applied impulsive load causes the inner surface of the cylinder wall to spall if the impulse intensity is sufficiently high. When spall occurs, the thickness of the residual cylinder wall is reduced and some of the momentum originally delivered to the wall is carried away by the spall. The auxiliary algorithms enable us to estimate the residual thickness of the wall, if spall occurs, and the momentum remaining in it. These are the required inputs for the algorithm of Equation 36. The auxiliary algorithms are intended to be used only if the actual applied impulse intensity is greater than 3.5.

Summarizing the problem defined so far, we have a basic algorithm which has four input parameters with associated uncertainty . The algorithm itself is also assumed to have associated uncertainty. Two of the parameters used in the algorithm must first be computed from auxiliary algorithms. We now describe in some detail how all these algorithms are made available to the code.

The basic algorithm, Equation 36, becomes available to the code via the user supplied subroutine TF. In order to prepare the required subroutine, we first must decide in what order we wish to refer to the uncertain parameters. This is necessary since the randomized value for each parameter used in each evaluation of the algorithm is stored in a particular location in an array called A. To code the basic algorithm in the present example analysis, we elect to associate the randomized value for ρ with array element $A(1)$, the randomized value for σ with $A(2)$, I_r with $A(3)$, and h_r with $A(4)$. To accomplish this association, we simply input the nominal values (and associated K-factors) for these parameters in this order when prompted for them during program execution. Since both I_r and h_r will be computed by the auxiliary algorithms, the nominal values we input for these two parameters can be anything (provided we don't intend to use the input nominals in the auxiliary algorithm subroutine). We typically input unity (1.0) as the nominal value for parameters such as I_r and h_r. The actual nominal values for ρ and σ are input. All of these nominal values are stored by BETAFACT, in the order of their entry, in array $ANOM$. During program execution, $ANOM(1)$ is used in the computation of $A(1)$ (the randomized parameter value), $ANOM(2)$ for $A(2)$, etc.

If the basic algorithm under consideration is assumed not to have associated uncertainty, then the nominal values described above are all the nominals expected by

the code. With specified algorithmic uncertainty, however, it is necessary to enter an additional nominal value (with a value which must be 1.0 in this case) and the associated K-factor specifications. In our example, this nominal value can be thought of as another parameter which is assigned to array element $A(5)$ (or $ANOM(5)$). If algorithmic uncertainty is to be applied in a BETAFACT analysis, its nominal value (1.0) and associated K-factors must be the last set of uncertain parameter nominal inputs to the code.

In most instances (though not in the present case) we use the values of independent parameters in the evaluation of the basic algorithm in subroutine TF. The values of these independent parameters are contained in array $XVAL$. The order of the independent parameter values in this array is determined by the order in which the independent parameter values are input at the appropriate prompt by the code. In the present example there are two independent parameters; the actual applied impulse intensity I which we choose to enter first and which is thus assigned to $XVAL(1)$ and the pre-spall thickness h which is stored in $XVAL(2)$. The entries in array $XVAL$ are not changed during the execution of BETAFACT unless the user selects an option which enables them to be altered interactively.

Once we have picked an ordering of the uncertain parameters and the independent parameters, we can then proceed to formulate subroutine TF. The listing of TF which codes the basic algorithm of this example problem is provided in Appendix A. The algorithm is simple and its coding is straightforward. To facilitate checking of the algorithm coding, we use local variable names in the subroutine mnemonic of the respective variables in the algorithm. For instance, we set $ARHO = A(1)$ since array element $A(1)$ is the randomized value of the density, $ASIG = A(2)$ which is the randomized ultimate stress, etc. (In the listing given for TF we also use a constant multiplier 1.0×10^6. This multiplier serves to convert impulse intensity I, input in ktaps, to taps for consistency with the units of the other algorithm input parameters.)

Subroutine AUXALGO, which contains the auxiliary algorithms, is prepared in much the same way as subroutine TF. A listing of the AUXALGO subroutine used in the present analysis is given in Appendix A. It is important to note that we use the nominal parameter value array $ANOM$ and not the array A (randomized parameter value array) in order to compute the auxiliary algorithms. Once again mnemonic local variable names are recommended to help the user understand and verify the coding of the auxiliary algorithms. For instance, in this subroutine we use $AI = XVAL(1)$ to locally represent the applied impulse intensity and $AH = XVAL(2)$ the original cylinder wall thickness.

After both subroutine TF and AUXALGO are prepared, we compile and link them to the BETAFACT object file to generate the executable version of the code which applies specifically to our problem. The actual VAX/VMS commands needed for compiling and linking the subroutines are described in Section 5. We can then proceed to run the code interactively.

The first example analysis we consider of the realistic algorithm described above is determination of the overall uncertainty associated with the output of the algo-

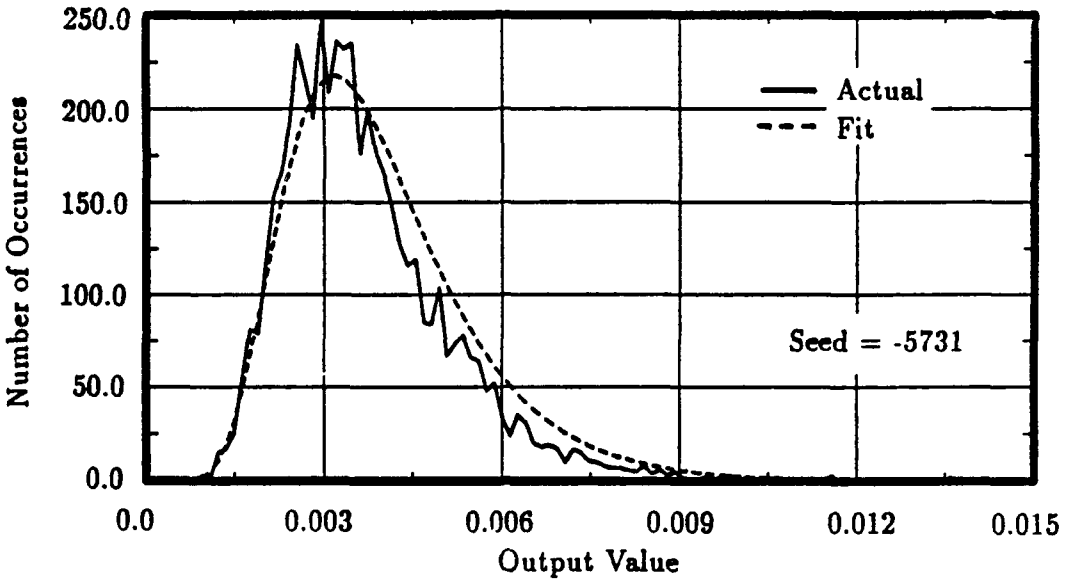

Figure 11. Realistic algorithm results distribution for original K-factor specifications and lognormally distributed uncertainty.

rithm. As specific input we use $\rho = 2.7$ gm/cm^3 with $K_\rho = 1.05, \sigma = 3.45 \times 10^9$ dyne/cm^2(50ksi) with $K_\sigma = 1.10, I_r$ (nominal) = 1.0 with $K_I = 1.25, h_r$ (nominal) = 1.0 with $K_h = 1.40$, and algorithm (nominal) = 1.0 with $K_{algo} = 1.25$. We choose to model all the uncertainties with lognormal distributions. For the independent parameters we specify I (applied) = 10.0 ktap and h(original) = 0.38cm. We note that the specifications for K_I and K_h are the K-factors associated with the outputs of the auxiliary algorithms, I_r and h_r, respectively. These K-factors represent the effective uncertainty associated with the auxiliary algorithms, including all important contributions to the overall algorithmic uncertainty. Two previous BETAFACT analyses, one for each auxiliary algorithm, are assumed to have been used to generate the combined uncertainty specification for each auxiliary algorithm.

Using the values described above, evaluating the algorithm 5000 times to generate the results distribution, and then computing the statistics of the distribution, we find that its mean is $\mu = 3.69 \times 10^{-3}$, its standard deviation is $\sigma = 1.42 \times 10^{-3}$ and the associated K-factor uncertainty is $K_\epsilon = 2.13$. A plot of the results distribution (continuous line curve) and fit to it (dash line curve) is shown in Figure 11. The theoretical mean for this problem is 3.47×10^{-3}, which is about 6% less than the computed mean.

A very useful feature of BETAFACT is that, once a problem has been fully defined for a set of input parameters and associated K-factors, it is easy to re-analyze the problem using varied parameter values and/or uncertainties. Thus we can use the code to perform algorithm sensitivity studies.

As a first example of such studies, we determine which source of uncertainty in our problem is the most significant contributor to the overall combined uncertainty. Clearly the most significant source is either the uncertainty associated with the basic algorithm or with one of the two auxiliary algorithms. If we re-analyze the problem described above, but with $K_I = 1.01$ (ie., very little uncertainty), we find that the code estimates the combined uncertainty to be $K = 1.77$; 17% less than K_ϵ given above. For $K_h = 1.01$ and all other inputs as originally specified, we find that $K = 1.68$; 21% less than K_ϵ. Finally, for $K_{algo} = 1.01$ and all other inputs with their original values, we compute $K = 2.02$; only 5% less than K_ϵ. Thus the uncertainty associated with the auxiliary algorithm which computes the cylinder wall thickness is the most significant, that associated with the residual impulse intensity algorithm only slightly less significant, and that attributed to the basic algorithm in relative terms almost insignificant. While this ordering of uncertainty significance may have been anticipated since K_h is numerically the largest of the three uncertainties considered, the greater exponent (2.0) on I_r in the basic algorithm compared to that of h_r (1.5) nearly reverses the ordering. A more complicated algorithm would make determination by inspection of the most significant uncertainty source a much more difficult task. However, it would be straightforward with BETAFACT.

Now suppose we are asked which of the following is more beneficial in reducing the overall combined uncertainty; reducing K_I from 1.25 (its original value) to 1.10 or reducing K_h from 1.40 (its original value) to 1.25? Further suppose the reductions will be achieved via test programs with the former estimated to cost \$200K and the latter \$100K. Which test program should be pursued? BETAFACT enables us to quantify justifiable answers to these questions. The code can thus be employed as a useful management and decision making tool.

First consider the overall uncertainty reduction obtained by reducing the contributing uncertainties. Re-analyzing the original problem with K_I reduced from 1.25 to 1.10, we find that the combined K-factor decreases from 2.13 to 1.83, a 14% reduction. If instead we reduce K_h from 1.40 to 1.25, we find that K_ϵ decreases from 2.13 to 1.88, a 12% reduction. Thus the maximum benefit is obtained by reducing K_I in this case.

Now consider the cost factor. On the one hand, reducing K_ϵ by reducing K_I costs, on the average, \$14.3K per percentage point. On the other, reducing K_ϵ by improving K_h costs only \$8.3K, on the average, per percentage point. Thus a percentage point improvement in the latter case costs only 58% what it does, on the average, in the former. From the viewpoint of maximized uncertainty reduction, we would recommend the option which reduces K_I. However, from a cost viewpoint, clearly it would be more economical to pursue the K_h reduction option.

Other sensitivity studies which can be accomplished using BETAFACT are probably evident to the reader. The above simple example clearly demonstrates that the code is adept at performing such quantitative analysis.

SECTION 5
BETAFACT USER'S MANUAL

The analysis of an algorithm with BETAFACT is a two step process; preparation of the problem definition (including formulation of user supplied subroutines) and analysis of the problem with the code. Since BETAFACT executes interactively, the bulk of the effort involved in analyzing a new algorithm is incurred in defining the problem.

In the following, we describe some preliminaries to the execution of the code (subsection 5.1), review the steps involved in formulating a problem definition (subsection 5.2) and describe in detail the interactive execution of BETAFACT (subsection 5.3). Many details of the problem definition phase are described more completely in previous sections of this report (especially Sections 3 and 4) and will be repeated only briefly here. The reader should refer to the previous report sections for the additional details.

5.1 PRELIMINARIES TO EXECUTION.

We first assume the user has installed the file containing the BETAFACT source code (named for reference here as BETAFACT.FOR) and the object file (BETAFACT.OBJ) on the computer to be used in the algorithm analyses. Installation here means simply having a copy of each file in the user's local file area. A listing of the source code is provided as Appendix B to this report. It can be used to verify the completeness of the user's source file.

If the user does not have file BETAFACT.OBJ available, but does have the source file, an object file can be generated with the command

FOR/CONTINUATIONS=99 BETAFACT.FOR

The above command applies to compilation of the code on VAX type computers operating under VMS. Since BETAFACT is coded using standard FORTRAN-77, probably it can be readily compiled on any computer (such as a PC) with a FORTRAN compiler. A command similar to the above would have to be issued on the non-VAX computer to generate the desired BETAFACT object file. The CONTINUATIONS parameter in the above command is necessary since some data statements in the BETAFACT source continue over 50 or more lines of coding.

In order to obtain an executable version of BETAFACT which is specifically for analysis of the user's algorithm, the algorithm must be made available to the code via the user supplied subroutine named TF (which stands for Transfer Function in the FAST terminology). If the basic algorithm under consideration happens to use auxiliary algorithms (i.e., algorithms which compute parameter values which are part of the input to the basic algorithm), these must be made available to the code via the

user supplied subroutine named AUXALGO (for auxiliary algorithm). These two subroutines contain the FORTRAN coding of the user's basic and auxiliary algorithms, respectively.

Once the user's subroutines are written and compiled, they must be linked to the BETAFACT object file as follows:

1. If auxiliary algorithms are used, then link with

LINK BETAFACT,TF,AUXALGO

2. If auxiliary algorithms are not used, then simply link with

LINK BETAFACT,TF

A nonfatal error message will be obtained with the second LINK above. This merely notifies the user that AUXALGO is referenced in BETAFACT but is not defined since no subroutine AUXALGO was linked. The code will execute properly even though this nonfatal error was encountered during linking.

The result of the above linking procedures is a file named BETAFACT.EXE which is the executable version of the code tailored specifically for analysis of the user's algorithm. To begin interactive execution of this program file, it is only necessary to issue the command (on VAX type computers).

RUN BETAFACT

From this point on, the user need only respond to the prompts issued by the code.

5.2 PROBLEM DEFINITION.

The user needs to understand three single index arrays and one variable used by BETAFACT in order to write useful user subroutines for the code. The three arrays are named $ANOM(i), XVAL(i)$ and $A(i)$ and the variable $YVAL$. The meaning of each in the code is as follows:

1. $ANOM(i)$ - This array contains the nominal values of the parameters with associated uncertainty which are used in the basic algorithm. Each source of uncertainty is identified with a specific entry in array $ANOM$. The user defines the association of an uncertain parameter with say the i-th entry in $ANOM$ by entering it as the i-th parameter value when the code prompts for nominal values of parameters with associated uncertainty. If the actual parameter nominal value is to be determined by an auxiliary algorithm in subroutine AUXALGO

(see subsection 4.2 for a specific example of this situation), the nominal value specified interactively for the parameter can be any value, provided the input value isn't used in AUXALGO. We recommend using a nominal value of 1.0 for such parameters. If the user's basic algorithm is to have an associated uncertainty (again see subsection 4.2 for an example), this constitutes a source of uncertainty which is treated in BETAFACT as a parameter with a nominal value which must be 1.0. This parameter nominal must also be the last uncertain parameter nominal value entered interactively by the user.

2. $XVAL(i)$ - This array contains the values of parameters used in the user's algorithms which do not have associated uncertainty. They are also referred to as independent variables or parameters. The value of the i-th independent parameter is associated with the i-th entry of array $XVAL$ by it being the i-th value entered at the BETAFACT prompt for the nominal values of independent parameters.

3. $A(i)$ - This array contains in its i-th location the current randomized value of an uncertain parameter corresponding to the nominal parameter value stored as the i-th entry in array $ANOM$. Section 3 describes how these randomized values are generated.

4. $YVAL$ - This variable has the value computed for the output of the user's basic algorithm for the current set of randomized input parameters but before basic algorithm uncertainty has been applied.

The arrays $ANOM$ and $XVAL$ are accessible in subroutine AUXALGO. Indeed, the purpose of AUXALGO is to assign one or more of the values of array $ANOM$. All three arrays can be used in subroutine TF. The user must be careful not to change any of the array entries (particularly those of $ANOM$ and $XVAL$) in that subroutine, however. The value of the evaluated algorithm returned by subroutine TF must be assigned to $YVAL$.

Listings of example TF and AUXALGO subroutines are given in Appendix A. The examples given are discussed in Section 4, particularly subsection 4.2. The example subroutines use mnemonic local variable names to code the algorithms. This facilitates understanding and identification of the algorithm and is a practice we recommend.

Once the user has developed an executable version of BETAFACT which is specific to the algorithm(s) in question, only one step remains before interactive execution of the problem can commence. This step entails simply the collection of all the parameter (both uncertain and independent) nominal values to be used in the analyses, definition of the K-factor uncertainties associated with each uncertain parameter, and selection of the probability distribution type which is to be used in BETAFACT to model uncertain parameter distributions. Any one of four types of distributions (normal, lognormal, Beta, and uniform) may be selected. The reader is cautioned that the meaning of a K-factor uncertainty specification changes from one probability

distribution type to another. Section 3 should be consulted for the precise meaning of K-factors as a function of distribution type.

With the above data and a problem specific executable version of BETAFACT in hand, interactive analysis of the subject algorithm may proceed.

5.3 INTERACTIVE EXECUTION.

In this subsection we describe the specific meaning of prompts issued by BETAFACT during the course of interactive execution. The meaning of the majority of these is self-evident or has been explained, perhaps obliquely, in earlier sections of this report. We discuss them here for completeness and to illustrate what is encountered during the proper execution of a BETAFACT analysis session.

We begin by first mentioning that much of what is entered interactively to BETAFACT is not forever lost once it scrolls off the user's monitor. Most input data are echoed not only to the monitor but also to unit 2 (file FOR002.DAT on VAX computers). This file remains after execution of BETAFACT is terminated. It can be edited and printed at will. Some data are written to unit 3 (FOR003.DAT) only. This data consists of the $x - y$ pairs, one pair per line, which enable plotting of computed and fitted algorithm results distributions exterior to BETAFACT. The file also contains the cumulative probability distribution computed from the algorithm results distribution. The file is generated only if the user requests plot file generation in response to the corresponding program prompt. Under usual circumstances, neither file is excessively large. If BETAFACT aborts during an interactive session, usually because of an invalid input to a prompt, any data previously written to unit 2 or 3 will usually still be accessible. This is important since no coding is implemented in the current version of BETAFACT which checks input as received and enables the user to modify it if it is incorrect and may lead to an error termination.

A BETAFACT interactive session begins with the user entering the following command (on VAX machines) and hitting return:

RUN BETAFACT

The program then begins to execute, issuing the following prompts (in some cases we abbreviate the prompts here) as it proceeds:

1. 'Does the algorithm to be evaluated have an overall uncertainty associated with its output? [Y/N]'

   If it does, the interactive user should type Y or y and hit return. If the basic algorithm does not have associated uncertainty, the user enters N or n (or anything else for that matter).

2. 'Are auxiliary algorithms required to determine nominal parameter values which are dependent on the values of the independent variables? [Y/N]'

Enter Y or y if subroutine AUXALGO is to be accessed, else enter N or n (or anything else).

3. 'Enter the number of parameters (*IPARAM*) in the algorithm which have associated uncertainty (include in the count whether or not the algorithm has an overall uncertainty), and select the distribution type for modeling the uncertainties (1,2,3 or 4 for normal, lognormal, Beta, and uniform, respectively).'

   Enter the number of parameters with associated uncertainty (number of actual parameters plus 1 if algorithm uncertainty is to be applied) followed by a space and then 1,2,3, or 4 to select a distribution type.

4. 'Enter the nominal values of the parameters which are normally (lognormally, Beta, or uniformly) distributed and the corresponding K-factors.'

   On a new line for each uncertain parameter enter its nominal value and K-factors (one or two required depending on distribution type) with each entry on the line separated by a space. If two K-factors are entered, enter the low-side K-factor first and then the high-side K-factor. Hit RETURN after each line of data.

   After the entire set of uncertain parameter nominal values are entered, the data are echoed to the monitor and execution proceeds.

5. 'Enter the number of independent variables (*IXVAL*) which appear in the algorithm or are used in *AUXALGO*.'

   Enter the number of independent parameters. At least one must be used. If such parameters do not occur naturally in an algorithm, one may be included as an algorithm multiplier with a value of unity.

6. 'Enter the values of the independent variables.'

   Enter the values, one after another, separated by a space. Continue on additional lines if necessary.

   After the entire set of independent variables are entered, the data are echoed to the monitor.

7. 'Enter the number of evaluations of the algorithm to be made for generating the results distribution and a negative seed integer for the random number generator.

   Enter a positive integer (say 5000), a space, and a negative integer (e.g., -5731) and then RETURN. The random number generator uses a negative seed integer. If the entered value is positive, the random number generator uses its negative value.

   If parameter uncertainties are to be modelled with Beta cumulative probability distributions, the following series of prompts are then issued:

8. (a) 'Select whether to use tabulated or calculated Beta cumulative probability distribution values; 1 - use tabulated, 2 - use calculated.'

Enter 1 or 2, as appropriate.

If 1 is entered, the program determines and reports the number of the tabulated distribution it will use to model the distribution associated with each uncertain parameter.

If 2 is entered, the program computes and reports allowable standard deviation boundaries (see Equation 21) for each uncertain parameter and prompts.

(b) 'Enter the standard deviations of the Beta distributed parameters.'

Enter the standard deviation for each parameter, separating by a space and using as many lines as required.

The code echoes the input values and reports the Beta cumulative probability distribution values (21 points) computed for each uncertain parameter.

The code then proceeds to evaluate the algorithm the requested number of times, computes the statistics of the results distribution, and then reports the mean, variance, standard deviation and K-factor(s) determined for the algorithm. It then prompts:

9. 'Do you want plot files of the results distribution and fit to be generated? [Y/N].'

   Enter Y or y if x-y plot data are to be saved, else enter N, n, or any other character.

Finally the code prompts:

10. 'Select option to quit or to run a modified version of the current problem (0,1,2,...,16).'

    A total of 17 options are available, one which terminates current execution and the other 16 which enable various degrees of problem modification and reanalysis. See the listing (subroutine OPTIONS) for the list of options available. In response to this prompt, type the number of the option selection and hit RETURN.

At this point, the program reissues one or more of the prompts described above and execution proceeds as before. Thus the above list of interactions of the user with BETAFACT is complete.

SECTION 6
CONCLUSIONS/RECOMMENDATIONS

In this report we have described a numerical approach for statistically combining many sources of algorithm uncertainty in order to quantify the overall uncertainty associated with an algorithm result. The interactive BETAFACT program was written to accomplish this task. The code allows the user to model uncertainty distributions using either normal, lognormal, Beta, or uniform probability distributions. We described in detail how the code can be used to analyze essentially any algorithm with uncorrelated parameter uncertainties. The only special skill required by the user to use BETAFACT to achieve this end is moderate competency in writing FORTRAN coding for the algorithms. During execution, the code prompts for all data required to process the coded algorithms.

We have used BETAFACT extensively in our LTH-3 Program lethality assessment activities. A code of this type seems to be essential in order to prepare algorithms and define associated uncertainties which are then used as input to the DNA FAST code. BETAFACT has also been shown to be a tool useful in program management and decision making processes.

The current version of BETAFACT has some limitations which in some applications could be significant. We list below some of these limitations and give our recommendations concerning whether or not they should be alleviated.

1. The chief limitation of the code is that sources of uncertainty are assumed to be uncorrelated. A substantial effort would be required to implement a correlated uncertainty capability in the code. Such an effort is recommended if there is a desire to enhance the generality of the code.

2. A possibly important limitation is the requirement that all sources of uncertainty in a given analysis must be modelled with one type of probability distribution (e.g., all normal or all lognormal, etc.). It would be straightforward to modify BETAFACT to allow modelling different sources of uncertainty with different probability distributions in the same analysis. The resulting algorithm results distributions would tend to be hybrid in this case and perhaps not easily or accurately modelled by one of the standard types of distributions. If a tabular form of the algorithm results distribution is useful (for instance, if FAST was modified to accept and use arbitrary probability distributions), then modification of BETAFACT to allow mixing of distribution types in an analysis is recommended.

3. Only four types of probability distributions (normal, lognormal, Beta, and generalized uniform) are available in BETAFACt. We recommend adding additional distribution types to the code. The effort involved would not be substantial.

SECTION 7
LIST OF REFERENCES

1. Coleman, H. W. and Steel, W. G., Experimentation and Uncertainty Analysis for Engineers, John Wiley and Sons, New York, 1989.

2. Rowan, W. H., "Failure Analysis by Statistical Techniques (FAST) Vol. I - User's Manual," DNA 3336F-1, TRW 24006-001-RU-00, TRW Systems Group, September 1974.

3. Simons, D.A., "Systematic Uncertainties in Solid Motor Case Design and Vulnerability," RDA-TR-147821-001, RDA Associates, June 1988.

4. APTEK, Inc., "LTH-3 FY88 Incremental Lethality Assessment," December 1988, Unpublished.

5. APTEK, Inc., "LTH-3 FY89 Incremental Lethality Assessment," February 1990, Unpublished.

APPENDIX A
EXAMPLE USER SUBROUTINES

1. This is the simple TF subroutine used in the verification analyses of the BETAFACT code. For a discussion of the simple algorithm and the analysis results, see subsection 4.1 of the BETAFACT documentation.

```
      SUBROUTINE TF(ANOM,A,XVAL,YVAL)
C
C      THIS EXAMPLE FOR SUBROUTINE TF CONTAINS THE FORTRAN CODING FOR
C      THE SIMPLE ALGORITHM CONSIDERED IN SUBSECTION 4.1 OF THE BETAFACT
C      CODE DOCUMENTATION:
C
C                                R = X
C
C      THE VARIABLE X, WHICH HAS ASSOCIATED UNCERTAINTY, HAS A
C      RANDOMIZED VALUE WHICH IS PASSED TO THIS SUBROUTINE AS THE
C      FIRST ELEMENT, A(1), IN ARRAY A.  THE LOCAL NAME FOR THIS
C      VARIABLE IS ADUM.  THE RANDOMIZED VARIABLE VALUE IS MULTIPLIED
C      BY THE VALUE OF THE INDEPENDENT VARIABLE (WHICH HAS VALUE 1.0)
C      WHICH IS STORED AS THE FIRST ELEMENT OF ARRAY XVAL.  THE
C      OUTPUT R OF THE SIMPLE ALGORITHM IS ASSIGNED TO YVAL, AS
C      REQUIRED BY BETAFACT.
C
      REAL*4    ANOM(*), A(*), XVAL(*)
      ADUM = A(1)
      YVAL = XVAL(1)*ADUM
      RETURN
      END
```

2. The following are the AUXALGO and TF subroutines used in the realistic algorithm example analyses described in subsection 4.2 of the BETAFACT documentation.

```
      SUBROUTINE AUXALGO(ANOM, XVAL)
C
C      THIS SAMPLE SUBROUTINE CONTAINS CODING NEEDED FOR EVALUATING THE
C      MODIFIED VERSION OF THE HKL ALGORITHM DESCRIBED IN SUBSECTION 4.2
C      OF THE BETAFACT DOCUMENTATION.
C
      REAL*4    ANOM(*), XVAL(*)
      AI = XVAL(1)
      AH = XVAL(2)
      IF (AI.LT.3.5) THEN
         ANOM(3) = AI
         ANOM(4) = AH
         RETURN
      ELSE
         FAC = SQRT(2.0*AH)
         ANOM(3) = (3.5 + 0.08*(AI - 3.5))*FAC
         ANOM(4) = AH - (0.05 + 0.001*(AI - 3.5))*FAC
         RETURN
      ENDIF
      RETURN
      END

      SUBROUTINE TF(ANOM,A,XVAL,YVAL)
C
C      THIS SAMPLE SUBROUTINE CONTAINS CODING FOR THE MODIFIED VERSION
C      OF THE HKL ALGORITHM DESCRIBED IN SUBSECTION 4.2 OF THE BETAFACT
C      DOCUMENTATION.
C
      REAL*4    ANOM(*), A(*), XVAL(*)
      ARHO = A(1)
      ASIG = A(2)
      AIR  = A(3)
      AHR  = A(4)
      YVAL = 0.5*AIR*AIR/(ARHO*ASIG*AHR**1.5)
      YVAL = YVAL*10**6.
      RETURN
      END
```

APPENDIX B
BETAFACT CODE LISTING

```
      PROGRAM BETAFACT
C
C     PROGRAM BETAFACT IS USED TO COMPUTE THE COMBINED UNCERTAINTY
C     TO BE ASSOCIATED WITH AN ALGORITHM WHICH IS A FUNCTION OF
C     SEVERAL PARAMETERS, EACH OF WHICH MAY HAVE ASSOCIATED
C     UNCERTAINTY.  THE ALGORITHM, ITSELF, MAY HAVE SPECIFIED
C     UNCERTAINTY ASSOCIATED WITH ITS OUTPUT, REGARDLESS OF
C     WHETHER OR NOT ITS PARAMETERS HAVE UNCERTAIN VALUES. ALL
C     THESE SOURCES OF UNCERTAINTY ARE ACCOUNTED FOR IN BETAFACT
C     IN ORDER TO OBTAIN THE FINAL UNCERTAINTY ESTIMATE FOR THE
C     EVALUATED ALGORITHM.  ALL UNCERTAINTIES USED IN THE PROGRAM
C     ARE SPECIFIED IN THE FORM OF EQUIVALENT K-FACTORS.
C
C     THE UNCERTAINTIES ASSOCIATED WITH PARAMETERS AND ALGORITHMS
C     MAY BE MODELLED AS NORMALLY, LOGNORMALLY, BETA, OR UNIFORMLY
C     DISTRIBUTED IN BETAFACT.  THE USER IS PROMPTED TO DEFINE,
C     INTERACTIVELY, THE REQUIRED DISTRIBUTION TYPES AND, IN FACT,
C     THE ENTIRE SPECIFICATION OF THE PROBLEM TO BE SOLVED.  IT IS
C     ONLY REQUIRED THAT THE USER PROVIDE A SUBROUTINE (CALLED TF)
C     TO THE PROGRAM WHICH CONTAINS THE CODING FOR THE ALGORITHM
C     WHICH IS TO BE EVALUATED.  THE USER MAY ALSO PROVIDE AN
C     AUXILIARY SUBROUTINE (CALLED AUXALGO) TO EVALUATE THE VALUES
C     OF PARAMETERS USED IN THE MAIN ALGORITHM WHICH ARE OBTAINABLE
C     FORM OTHER SIMPLE ALGORITHMS OR RELATIONS.
C
C
C     BETAFACT WAS DEVELOPED ON A VAX TYPE COMPUTER OPERATING UNDER VMS.
C     TO USE BETAFACT, THE USER FIRST MUST COMPILE THE SUBROUTINE TF
C     (AND SUBROUTINE AUXALGO IF THERE IS ONE) AND THEN LINK BOTH TO
C     THE BETAFACT OBJECT FILE AS FOLLOWS:
C
C               LINK BETAFACT,TF,AUXALGO(if there is one)
C
C     THE RESULT OF THIS PROCESS IS A FILE CALLED BETAFACT.EXE, WHICH
C     IS THE BETAFACT EXECUTABLE FILE.  ONCE THIS FILE IS CREATED,
C     ITS EXECUTION IS ACHIEVED SIMPLY BY TYPING
C
C               RUN BETAFACT
C
C     HITTING RETURN, AND THEN ANSWERING THE PROMPTS ISSUED BY THE
C     PROGRAM.
```

```
C
C      A SIGNIFICANT PORTION OF BETAFACT IS FLEXIBLY DIMENSIONED.
C      ONLY THE ARRAYS EXPLICITLY RELATED TO THE LIBRARY OF 81
C      BETA CUMULATIVE PROBABILITY DISTRIBUTIONS USED BY THE
C      CODE HAVE FIXED DIMENSIONS.  IF THE NUMBER OF LIBRARY
C      BETA CUMULATIVE PROBABILITY DISTRIBUTIONS NEEDS TO BE
C      INCREASED, THE DIMENSIONS OF THE FOLLOWING ARRAYS WILL
C      HAVE TO BE CHANGED: BD, DBA, DBB, AMED, ALPH, AND BET.
C
C      ALL OTHER ARRAYS USED BY THE PROGRAM ARE DIMENSIONED VIA
C      A PARAMETER STATEMENT WHICH DEFINES THE FOLLOWING INTEGER
C      VARIABLES:
C
C        MAXPRM - MAXIMUM NUMBER OF VARIABLES WITH UNCERTAINTY.
C        MAXVAR - MAXIMUM NUMBER OF INDEPENDENT PARAMETERS.
C        MAXSIZ - MAXIMUM NUMBER OF ALGORITHM EVALUATIONS.
C        MAXINT - MAXIMUM NUMBER OF INTERVALS USED TO CALCULATE BETA
C                 CUMULATIVE PROBABILITY DISTRIBUTIONS.
C        MAXIN1 - MAXIMUM NUMBER OF INTERVALS USED TO DEFINE HISTOGRAMS.
C
C      OTHER PRINCIPAL INTEGER VARIABLES AND CONTROL PARAMETERS
C      USED BY THE CODE ARE AS FOLLOWS:
C
C        IPARAM - NUMBER OF PARAMETERS WITH UNCERTAINTY.
C        IPARA  - EQUAL TO IPARAM IF ALGORITHM DOES NOT HAVE AN OVERALL
C                 SPECIFIED UNCERTAINTY; ELSE IPARA = IPARAM - 1.
C        ITYPE  - DISTRIBUTION TYPE USED TO MODEL UNCERTAINTIES:
C                    = 1 NORMAL DISTRIBUTION.
C                    = 2 LOGNORMAL DISTRIBUTION.
C                    = 3 BETA DISTRIBUTION.
C                    = 4 UNIFORM DISTRIBUTION.
C        ISIZE  - NUMBER OF ALGORITHM EVALUATIONS TO BE MADE.
C        ISEED  - SEED (NEGATIVE AND ODD) USED TO INITIALIZE THE RANDOM
C                 NUMBER GENERATOR.
C        IALGO  - FLAG INDICATING SPECIFIED ALGORITHMIC UNCERTAINTY:
C                    = 1 ALGORITHM HAS OVERALL UNCERTAINTY.
C                    = ANYTHING ELSE; DOESN'T HAVE OVERALL UNCERTAINTY.
C        IXVAL  - NUMBER OF INDEPENDENT PARAMETERS USED IN ALGORITHM.
C                 THESE DO NOT HAVE ASSOCIATED UNCERTAINTY.
C        ITABL  - FLAG DEFINING WHETHER TABULATED OR CALCULATED BETA
C                 CUMULATIVE PROBABILITY DISTRIBUTIONS ARE TO BE USED:
C                    = 1 USE TABULATED DISTRIBUTIONS.
C                    = 2 USE CALCULATED DISTRIBUTIONS.
C        IXALG  - FLAG WHICH SPECIFIES WHETHER OR NOT AUXILIARY
C                 ALGORITHMS IN USER SUPPLIED SUBROUTINE AUXALGO
C                 ARE TO BE ACCESSED: IF = 1, THEN SUBROUTINE IS
```

```
C                ACCESSED, OTHERWISE NOT ACCESSED.
C        IOPT    - ANALYSIS MODIFICATION OPTIONS FLAG.  SEE SUBROUTINE
C                  OPTIONS FOR AVAILABLE OPTIONS.
C        NINTV   - NUMBER OF INTERVALS USED TO CALCULATE BETA
C                  CUMULATIVE PROBABILITY DISTRIBUTIONS.  RECOMMENDED
C                  NUMBER IS DEFAULT NINTV = MAXINT.
C        NINC    - NUMBER OF INTERVALS USED TO GENERATE HISTOGRAMS.
C                  RECOMMENDED NUMBER IS DEFAULT NINC = MAXIN1.
C        NINP    - UNIT NUMBER FOR INTERACTIVE INPUT (DEFAULT = 5).
C        NOUT    - UNIT NUMBER FOR TERMINAL SCREEN OUTPUT (DEFAULT = 6).
C        NPRT    - UNIT NUMBER FOR PRINTABLE OUTPUT (DEFAULT = 2).
C        NPLT    - UNIT NUMBER FOR PLOT FILE (HISTOGRAM) OUTPUT (DEFAULT = 3).
C
C      THE PRINCIPAL REAL ARRAYS AND VARIABLES USED BY THE PROGRAM ARE:
C
C        ANOM    - ARRAY OF NOMINAL VALUES OF PARAMETERS WITH UNCERTAINTY.
C        AKLO    - ARRAY OF PARAMETER LOW-SIDE K-FACTOR UNCERTAINTIES.
C        AKHI    - ARRAY OF PARAMETER HIGH-SIDE K-FACTOR UNCERTAINTIES.
C        ALO     - ARRAY OF PARAMETER LOW-SIDE EXTREME VALUES.
C        AHI     - ARRAY OF PARAMETER HIGH-SIDE EXTREME VALUES.
C        A       - ARRAY OF RANDOM VALUES OF PARAMETERS FOR A SINGLE
C                  ALGORITHM EVALUATION LOOP.
C        XVAL    - ARRAY OF INDEPENDENT PARAMETER VALUES.
C        YVAL    - RESULT OBTAINED IN SINGLE EVALUATION OF ALGORITHM.
C        RESULT  - ARRAY OF ALL COMPUTED YVAL RESULTS.
C        ASIG    - ARRAY OF SPECIFIED BETA DISTRIBUTION STANDARD DEVIATIONS;
C                  USED TO CALCULATE BETA DISTRIBUTIONS.
C        BD      - ARRAY OF LIBRARY OR CALCULATED BETA CUMULATIVE
C                  PROBABILITY DISTRIBUTIONS.
C        ALPHA   - BETA DISTRIBUTION ALPHA PARAMETER.
C        BETA    - BETA DISTRIBUTION BETA PARAMETER.
C        ALPH    - ARRAY OF ALPHA PARAMETER VALUES USED TO GENERATE THE
C                  LIBRARY OF BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C        BET     - ARRAY OF BETA PARAMETER VALUES USED TO GENERATE THE
C                  LIBRARY OF BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C        AMED    - ARRAY OF MEDIAN (50%) VALUES OF THE LIBRARY OF
C                  BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C        RN      - VARIABLE EQUAL TO THE MOST RECENT RANDOM NUMBER.
C        AMEAN   - MEAN OF THE ALGORITHM RESULTS DISTRIBUTION.
C        VAR     - VARIANCE OF THE ALGORITHM RESULTS DISTRIBUTION.
C        SIGMA   - STANDARD DEVIATION OF ALGORITHM RESULTS DISTRIBUTION.
C        AKEST   - COMBINED EFFECTIVE K-FACTOR ESTIMATE FOR CASES OF
C                  NORMAL AND LOGNORMAL MODELLED UNCERTAINTIES.
C        AKEST1  - LOW-SIDE COMBINED EFFECTIVE K-FACTOR ESTIMATE FOR
C                  BETA AND UNIFORM MODELLED UNCERTAINTIES.
C        AKEST2  - HIGH-SIDE COMBINED EFFECTIVE K-FACTOR ESTIMATE FOR
```

```
C                BETA AND UNIFORM MODELLED UNCERTAINTIES.
C        AKAVE  - K-FACTOR EQUAL TO THE AVERAGE OF AKEST1 AND AKEST2.
C
C      THE PRINCIPAL INTEGER ARRAYS USED BY THE PROGRAM ARE:
C
C        IBD    - ARRAY OF THE BETA CUMULATIVE PROBABILITY DISTRIBUTION
C                 NUMBER ASSOCIATED WITH EACH BETA DISTRIBUTED PARAMETER.
C        IFREQ  - FREQUENCY OF OCCURRENCE ARRAY USED TO ACCUMULATE
C                 HISTOGRAM DATA.
C
      PARAMETER (MAXPRM=20,  MAXVAR=10, MAXSIZ=10000,
     1           MAXINT=100, MAXIN1=100)
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      COMMON /BLK02/ BD(21,81)
      REAL*4    ANOM(MAXPRM),   ALO(MAXPRM),   AHI(MAXPRM)
      REAL*4    AKLO(MAXPRM),   AKHI(MAXPRM),  A(MAXPRM)
      REAL*4    XVAL(MAXVAR),   RESULT(MAXSIZ)
      REAL*4    ASIG(2,MAXPRM), AREA(MAXINT),  Y(MAXINT+1)
      REAL*4    XINT(MAXIN1+1), XOUT(MAXIN1)
      REAL*4    XXLN(MAXIN1+1), XXII(MAXIN1)
      INTEGER*4 IBD(MAXPRM), IFREQ(MAXIN1)
C
C      CONTROL PARAMETER 'IOPT' INITIALLY SET EQUAL TO 16
C      SINCE A COMPLETELY NEW PROBLEM IS TO BE DEFINED.
C
      IOPT  = 16
      NINC  = MAXINT
      NINTV = MAXIN1
C
C      ASSIGN DEFAULT UNIT NUMBERS FOR INPUT, OUTPUT, ETC:
C
C             NINP - UNIT NUMBER FOR INPUT FROM TERMINAL.
C             NOUT - UNIT NUMBER FOR OUTPUT TO TERMINAL.
C             NPLT - UNIT NUMBER FOR PLOT DATA.
C             NPRT - UNIT NUMBER FOR HARDCOPY OUTPUT.
C
      NINP  = 5
      NOUT  = 6
      NPLT  = 3
      NPRT  = 2
C
C      ASSIGN DEFAULT VALUES FOR OTHER CONTROL PARAMETERS.
C
      IALGO = 0
      IXALG = 0
```

```
      IXVAL = 1
      ITABL = 1
C
C      CALL SUBROUTINE CONTRL TO INPUT PROBLEM DEFINITION AND
C      MAIN CONTROL PARAMETERS.
C
   10 CALL CNTRL(ANOM, ALO, AHI, AKLO, AKHI, XVAL, ASIG, Y, AREA, IBD)
C
C      CALL SUBROUTINE SOLVE TO EVALUATE THE PROBLEM ALGORITHM.
C
      CALL SOLVE(ANOM, ALO, AHI, AKLO, AKHI, XVAL, A, IBD, RESULT)
C
C      CALL SUBROUTINE SORT TO ORDER THE ALGORITHM RESULTS
C      DISTRIBUTION FROM LOWEST TO HIGHEST VALUE.
C
      CALL SORT(RESULT,ISIZE)
C
C      CALL SUBROUTINE STATISTIC TO EVALUATE STATISTICS FOR THE
C      ORDERED RESULTS DISTRIBUTION, TO ESTIMATE OVERALL ALGORITHM
C      UNCERTAINTY, AND TO GENERATE PLOT FILES CONTAINING HISTOGRAM
C      DATA AND FITS TO THE HISTOGRAM DATA.
C
      CALL STATISTIC(RESULT, XINT, XOUT, XXLN, XXII, IFREQ)
C
C      CALL SUBROUTINE OPTIONS TO MODIFY THE CURRENT PROBLEM
C      DEFINITION WITHOUT EXITING FROM BETAFACT.
C
      CALL OPTIONS
C
C      TERMINATE THE PRESENT ANALYSIS ONLY IF IOPT = 0.
C
      IF (IOPT.NE.0) GO TO 10
      END
```

```
      SUBROUTINE CNTRL(ANOM, ALO, AHI, AKLO, AKHI, XVAL, ASIG, Y,
     1                 AREA, IBD)
C
C       SUBROUTINE CNTRL CONTROLS THE INPUT OF THE DATA WHICH DEFINE A
C       PROBLEM OR MODIFICATIONS TO A PREVIOUSLY DEFINED PROBLEM.  THE
C       SUBROUTINE INTERACTIVELY PROMPTS THE USER TO SPECIFY THE VALUES
C       OF THE FOLLOWING CONTROL PARAMETERS AND INTEGER VARIABLES:
C
C         IALGO  - FLAG IDENTIFYING USE OF OVERALL ALGORITHM UNCERTAINTY.
C         IXALG  - FLAG CONTROLLING ACCESSING OF AUXILIARY ALGORITHMS.
C         IPARAM - NUMBER OF PARAMETERS WITH UNCERTAINTY.
C         ITYPE  - DISTRIBUTION TYPE FOR MODELLING UNCERTAINTIES.
C         ISIZE  - NUMBER OF REQUESTED ALGORITHM EVALUATIONS.
C         ISEED  - SEED INTEGER (<0 AND ODD) FOR RANDOM NUMBER GENERATOR.
C
C       WHETHER OR NOT ONE OR ALL OF THE ABOVE INTEGER VARIABLES NEEDS
C       TO BE ENTERED IS DETERMINED BY THE CURRENT VALUE OF THE OPTIONS
C       CONTROL PARAMETER IOPT.
C
C       SUBROUTINE CNTRL ALSO CALLS TWO SUBROUTINES:
C
C          SUBROUTINE INPUT      - FOR ENTERING PARAMETER NOMINAL VALUES,
C                                  ASSOCIATED UNCERTAINTIES, AND THE VALUES
C                                  OF INDEPENDENT PARAMETERS WHICH DO NOT
C                                  HAVE ASSOCIATED UNCERTAINTY.
C          SUBROUTINE ASSIGNBETA - WHICH CONTROLS ASSOCIATION OF A SPECIFIC
C                                  TABULATED OR COMPUTED BETA CUMULATIVE
C                                  PROBABILITY DISTRIBUTION WITH PARAMETERS
C                                  HAVING BETA DISTRIBUTED UNCERTAINTY.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      REAL*4    ANOM(*), ALO(*), AHI(*), AKLO(*), AKHI(*), XVAL(*)
      REAL*4    ASIG(2,*), Y(*), AREA(*)
      INTEGER*4 IBD(*)
      CHARACTER*1 ANS1
C
C       THE VALUE OF THE OPTION CONTROL PARAMETER 'IOPT'
C       EQUALS 16 ONLY IF THIS IS A NEW PROBLEM OR THE
C       CURRENT PROBLEM IS TO BE COMPLETELY MODIFIED.
C       FOR IOPT = 16, THE CONTROL PARAMETERS 'IALGO' AND
C       'IXALG' NEED TO BE SPECIFIED.
C
      IF (IOPT.NE.16) GO TO 10
C
C       SPECIFY CONTROL PARAMETER 'IALGO' WHICH FLAGS WHETHER OR NOT
```

```
C      THE ALGORITHM HAS AN OVERALL ASSOCIATED UNCERTAINTY.
C      IF IALGO = 1, THEN THE ALGORITHM HAS OVERALL UNCERTAINTY.
C      OTHERWISE IT DOES NOT.
C
       WRITE(NOUT,1000)
 1000  FORMAT(/,' DOES THE ALGORITHM TO BE EVALUATED HAVE AN OVERALL',/,
     1         ' UNCERTAINTY ASSOCIATED WITH ITS OUTPUT? [Y/N]',/)
       READ(NINP,1001) ANS1
 1001  FORMAT(A1)
       IF ((ANS1.EQ. 'Y').OR.(ANS1.EQ. 'y')) IALGO = 1
C
C      SPECIFY CONTROL PARAMETER 'IXALG' WHICH FLAGS WHETHER OR NOT
C      THE ALGORITHM MUST ACCESS AUXILIARY ALGORITHMS TO OBTAIN VALUES
C      FOR SOME OF ITS PARAMETERS.  IF IXALG = 1, THEN AUXILIARY
C      ALGORITHMS, CODED IN THE USER SUPPLIED SUBROUTINE AUXALGO,
C      WILL BE ACCESSED.  OTHERWISE AUXILIARY ALGORITHMS ARE NOT
C      USED.
C
       WRITE(NOUT,1100)
 1100  FORMAT(/,' ARE AUXILIARY ALGORITHMS REQUIRED TO DETERMINE',/,
     1         ' NOMINAL PARAMETER VALUES WHICH ARE DEPENDENT ON',/,
     2         ' THE VALUES OF THE INDEPENDENT VARIABLES? [Y/N]',/)
       READ(NINP,1001) ANS1
       IF ((ANS1.EQ. 'Y').OR.(ANS1.EQ. 'y')) IXALG = 1
C
C      IF THIS IS A MODIFICATION ANALYSIS, CHECK THE VALUE OF 'IOPT'
C      TO DETERMINE IF THE NUMBER OF ALGORITHM PARAMETERS AND/OR
C      DISTRIBUTION TYPE SPECIFICATION NEED TO BE CHANGED.
C
    10 IF ((IOPT.EQ.1).OR.(IOPT.EQ.2).OR.(IOPT.EQ.3).OR.(IOPT.EQ.5).OR.
     1    (IOPT.EQ.6).OR.(IOPT.EQ.8).OR.(IOPT.EQ.11)) GO TO 20
C
C      SPECIFY THE NUMBER 'IPARAM' OF ALGORITHM PARAMETERS WHICH
C      HAVE ASSOCIATED UNCERTAINTY AND THE DISTRIBUTION TYPE 'ITYPE'
C      TO BE USED TO MODEL THE PARAMETER UNCERTAINTIES.
C
       WRITE(NOUT,1200)
 1200  FORMAT(/,' ENTER THE NUMBER OF PARAMETERS (IPARAM) IN THE',/,
     1         ' ALGORITHM WHICH HAVE ASSOCIATED UNCERTAINTY ',/,
     2         ' (INCLUDE IN THE COUNT WHETHER OR NOT THE ALGORITHM ',/,
     3         ' HAS AN OVERALL UNCERTAINTY), AND SELECT THE ',/,
     4         ' DISTRIBUTION TYPE FOR MODELLING THE UNCERTAINTIES ',/,
     5         ' FROM THE FOLLOWING (ENTER 1, 2, 3, OR 4):',/,/,/,
     6         '        1 - NORMAL DISTRIBUTION.',/,
     7         '        2 - LOGNORMAL DISTRIBUTION.',/,
     8         '        3 - BETA DISTRIBUTION.',/,
```

```
     9         '         4 - UNIFORM DISTRIBUTION.',/)
       READ(NINP,*) IPARAM, ITYPE
    20 IF (IOPT.EQ.2) GO TO 30
C
C      CALL SUBROUTINE INPUT TO ENTER DEPENDENT PARAMETER NOMINAL VALUES,
C      ASSOCIATED K-FACTOR UNCERTAINTIES, AND THE VALUE(S) OF THE
C      INDEPENDENT PARAMETER(S) FOR WHICH THE ALGORITHM IS TO BE
C      EVALUATED.
C
       CALL INPUT(ANOM, ALO, AHI, AKLO, AKHI, XVAL)
C
C      IF THIS IS A MODIFICATION ANALYSIS, CHECK THE VALUE OF 'IOPT'
C      TO DETERMINE IF THE NUMBER 'ISIZE' OF EVALUATIONS OF THE
C      ALGORITHM AND/OR THE SEED INTEGER 'ISEED' FOR THE RANDOM
C      NUMBER GENERATOR NEED TO BE CHANGED.
C
    30 IF ((IOPT.EQ.1).OR.(IOPT.EQ.3).OR.(IOPT.EQ.4).OR.(IOPT.EQ.6).OR.
     1    (IOPT.EQ.7).OR.(IOPT.EQ.10).OR.(IOPT.EQ.13)) GO TO 40
C
C      SPECIFY THE NUMBER 'ISIZE' OF ALGORITHM EVALUATIONS TO BE
C      PERFORMED AND A SEED INTEGER 'ISEED' FOR THE RANDOM NUMBER
C      GENERATOR.
C
       WRITE(NOUT,1300)
 1300 FORMAT(/,' ENTER THE NUMBER OF EVALUATIONS (ISIZE) OF THE',/,
     1         ' ALGORITHM TO BE MADE FOR GENERATING THE RESULTS',/,
     2         ' DISTRIBUTION AND A NEGATIVE SEED INTEGER (ISEED)',/,
     3         ' FOR THE RANDOM NUMBER GENERATOR.',/)
       READ(NINP,*) ISIZE, ISEED
    40 CONTINUE
C
C      CALL SUBROUTINE ASSGNBETA IF THE PRESENT ANALYSIS INVOLVES
C      MODELLING UNCERTAINTIES WITH CUMULATIVE BETA DISTRIBUTIONS.
C
       IF ((IOPT.EQ.1).OR.(IOPT.EQ.2).OR.(IOPT.EQ.5)) GO TO 50
       IF (ITYPE.EQ.3) THEN
          CALL ASSGNBETA(ANOM, ALO, AHI, AKLO, AKHI, ASIG, Y, AREA, IBD)
       ENDIF
    50 CONTINUE
       RETURN
       END
```

```
      SUBROUTINE INPUT(ANOM, ALO, AHI, AKLO, AKHI, XVAL)
C
C      SUBROUTINE INPUT INTERACTIVELY READS USER SUPPLIED DATA WHICH
C      DEFINE THE NOMINAL VALUES OF ALGORITHM PARAMETERS WITH UNCERTAINTY,
C      THE ASSOCIATED K-FACTOR UNCERTAINTIES, AND THE VALUES OF INDEPENDENT
C      PARAMETERS WHICH DO NOT HAVE ASSOCIATED UNCERTAINTY.  THE NUMBER
C      OF THE LATTER TYPE OF PARAMETERS, IXVAL, IS PROMPTED FOR BY THE
C      SUBROUTINE.  THE PARAMETER NOMINAL VALUES AND K-FACTOR UNCERTAINTIES
C      ARE USED IN THE SUBROUTINE TO COMPUTE THE EXTREME (LOW AND HIGH)
C      VALUES CORRESPONDING TO EACH PARAMETER WITH UNCERTAINTY.
C
C      THE PRINCIPAL ARRAYS DEFINED IN SUBROUTINE INPUT ARE AS FOLLOWS:
C
C        ANOM - NOMINAL VALUES OF THE PARAMETERS WITH UNCERTAINTY.
C        AKLO - LOW-SIDE K-FACTOR UNCERTAINTIES FOR THE PARAMETERS.
C        AKHI - HIGH-SIDE K-FACTOR UNCERTAINTIES FOR THE PARAMETERS.
C        ALO  - LOW-SIDE OF THE RANGE FOR EACH PARAMETER.
C        AHI  - HIGH-SIDE OF THE RANGE FOR EACH PARAMETER.
C        XVAL - VALUES OF THE INDEPENDENT PARAMETERS (NO UNCERTAINTY).
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      REAL*4    ANOM(*), ALO(*), AHI(*), AKLO(*), AKHI(*), XVAL(*)
C
C        DETERMINE IF IOPT VALUE REQUIRES INPUT OF PARAMETER DATA.
C
      IF ((IOPT.EQ.1).OR.(IOPT.EQ.2).OR.(IOPT.EQ.5)) GO TO 90
C
C        ENTER DATA FOR PARAMETERS WHICH ARE NORMALLY DISTRIBUTED.
C
      IF (ITYPE.EQ.1) THEN
        WRITE(NOUT,1000)
           DO 10 I = 1, IPARAM
              READ(NINP,*) ANOM(I), AKLO(I)
              AHI(I) = ANOM(I)*AKLO(I)
              ALO(I) = ANOM(I) - (AHI(I) - ANOM(I))
   10      CONTINUE
C
C        ECHO PARAMETER INPUT DATA AND COMPUTED RANGE.
C
        WRITE(NPRT,1100)
        WRITE(NOUT,1100)
           DO 20 I = 1, IPARAM
              WRITE(NPRT,1200) I, ANOM(I), AKLO(I), ALO(I), AHI(I)
              WRITE(NOUT,1200) I, ANOM(I), AKLO(I), ALO(I), AHI(I)
```

```
   20      CONTINUE
C
C        ENTER DATA FOR PARAMETERS WHICH ARE LOGNORMALLY DISTRIBUTED.
C
      ELSE IF (ITYPE.EQ.2) THEN
        WRITE(NOUT,1300)
           DO 30 I = 1, IPARAM
              READ(NINP,*) ANOM(I), AKLO(I)
              ALO(I) = ANOM(I)/AKLO(I)
              AHI(I) = ANOM(I)*AKLO(I)
   30      CONTINUE
C
C        ECHO PARAMETER INPUT DATA AND COMPUTED RANGE.
C
        WRITE(NPRT,1400)
        WRITE(NOUT,1400)
           DO 40 I = 1, IPARAM
              WRITE(NPRT,1500) I, ANOM(I), AKLO(I), ALO(I), AHI(I)
              WRITE(NOUT,1500) I, ANOM(I), AKLO(I), ALO(I), AHI(I)
   40      CONTINUE
C
C        ENTER DATA FOR PARAMETERS WHICH ARE BETA DISTRIBUTED.
C
      ELSE IF (ITYPE.EQ.3) THEN
        WRITE(NOUT,1600)
           DO 50 I = 1, IPARAM
              READ(NINP,*) ANOM(I), AKLO(I), AKHI(I)
              ALO(I) = ANOM(I)/AKLO(I)
              AHI(I) = ANOM(I)*AKHI(I)
   50      CONTINUE
C
C        ECHO PARAMETER INPUT DATA AND COMPUTED RANGE.
C
        WRITE(NPRT,1700)
        WRITE(NOUT,1700)
           DO 60 I = 1, IPARAM
              WRITE(NPRT,1800) I, ANOM(I),AKLO(I),AKHI(I),ALO(I),AHI(I)
              WRITE(NOUT,1800) I, ANOM(I),AKLO(I),AKHI(I),ALO(I),AHI(I)
   60      CONTINUE
C
C        ENTER DATA FOR PARAMETERS WHICH ARE UNIFORMLY DISTRIBUTED.
C
      ELSE IF (ITYPE.EQ.4) THEN
        WRITE(NOUT,1900)
           DO 70 I = 1, IPARAM
              READ(NINP,*) ANOM(I), AKLO(I), AKHI(I)
```

```
            ALO(I) = ANOM(I)/AKLO(I)
            AHI(I) = ANOM(I)*AKHI(I)
 70       CONTINUE
C
C        ECHO PARAMETER INPUT DATA AND COMPUTED RANGE.
C
         WRITE(NPRT,2000)
         WRITE(NOUT,2000)
         DO 80 I = 1, IPARAM
           WRITE(NPRT,2100) I, ANOM(I),AKLO(I),AKHI(I),ALO(I),AHI(I)
           WRITE(NOUT,2100) I, ANOM(I),AKLO(I),AKHI(I),ALO(I),AHI(I)
 80    CONTINUE
       ENDIF
C
C      CHECK THE VALUE OF 'IOPT' TO DETERMINE IF THE NUMBER AND/OR
C      VALUES OF THE INDEPENDENT PARAMETERS USED BY THE ALGORITHM
C      NEED TO BE ALTERED.
C
 90 IF ((IOPT.EQ.2).OR.(IOPT.EQ.3).OR.(IOPT.EQ.4).OR.(IOPT.EQ.8).OR.
    1    (IOPT.EQ.9).OR.(IOPT.EQ.10).OR.(IOPT.EQ.14)) GO TO 110
C
C      SPECIFY THE NUMBER 'IXVAL' OF INDEPENDENT PARAMETERS WHICH
C      APPEAR IN THE ALGORITHM.
C
       WRITE(NOUT,2200)
       READ(NINP,*) IXVAL
C
C      ENTER THE VALUES OF THE 'IXVAL' INDEPENDENT PARAMETERS.
C
       WRITE(NOUT,2300)
       READ(NINP,*) (XVAL(I), I = 1, IXVAL)
C
C      ECHO THE INDEPENDENT PARAMETER VALUES.
C
       WRITE(NPRT,2400)
       WRITE(NOUT,2400)
       DO 100 I = 1, IXVAL
          WRITE(NPRT,2500) I, XVAL(I)
          WRITE(NOUT,2500) I, XVAL(I)
100 CONTINUE
110 CONTINUE
       RETURN
1000 FORMAT(/,' ENTER THE NOMINAL VALUES OF THE PARAMETERS ',/,
     1         ' WHICH ARE NORMALLY DISTRIBUTED AND THE ',/,
     2         ' CORRESPONDING K-FACTOR.',/)
1100 FORMAT(/,' PARAMETER    NOMINAL   K-FACTOR  95% EXTREMES',/)
```

```
1200 FORMAT(1X,I3,5X,1PE10.4,2X,0PF10.3,2X,1PE10.4,2X,1PE10.4)
1300 FORMAT(/,' ENTER THE NOMINAL VALUES OF THE PARAMETERS ',/,
    1         ' WHICH ARE LOGNORMALLY DISTRIBUTED AND THE ',/,
    2         ' CORRESPONDING K-FACTOR.',/)
1400 FORMAT(/,' PARAMETER    NOMINAL   K-FACTOR  95% EXTREMES',/)
1500 FORMAT(1X,I3,5X,1PE10.4,2X,0PF10.3,2X,1PE10.4,2X,1PE10.4)
1600 FORMAT(/,' ENTER THE NOMINAL VALUES OF THE PARAMETERS ',/,
    1         ' WHICH ARE BETA DISTRIBUTED AND THE K-FACTORS,',/,
    2         ' (BOTH K-LOW AND K-HIGH, EVEN IF EQUAL, MUST',/,
    3         ' BE ENTERED).',/)
1700 FORMAT(/,' PARAMETER    NOMINAL      K-FACTORS',
    1         '       EXTREMES',/)
1800 FORMAT(1X,I3,5X,1PE10.4,2X,0PF10.3,2X,0PF10.3,2X,1PE10.4,
    1       1X,1PE10.4)
1900 FORMAT(/,' ENTER THE NOMINAL VALUES OF THE UNIFORMLY ',/,
    1         ' DISTRIBUTED PARAMETERS AND THE K-FACTORS,',/,
    2         ' (ENTER BOTH K-LOW AND K-HIGH).',/)
2000 FORMAT(/,' PARAMETER    NOMINAL      K-FACTORS',
    1         '       EXTREMES',/)
2100 FORMAT(1X,I3,5X,1PE10.4,2X,0PF10.3,2X,0PF10.3,2X,1PE10.4,
    1       1X,1PE10.4)
2200 FORMAT(/,' ENTER THE NUMBER OF INDEPENDENT VARIABLES (IXVAL)',/,
    1         ' WHICH APPEAR IN THE ALGORITHM.',/)
2300 FORMAT(/,' ENTER THE VALUES OF THE INDEPENDENT VARIABLES.',/)
2400 FORMAT(/,' INDEPENDENT                ',/,
    1         '  PARAMETER          VALUE ',/)
2500 FORMAT(5X,I5,5X,1PE12.4)
     END
```

```
      SUBROUTINE ASSGNBETA(ANOM, ALO, AHI, AKLO, AKHI, ASIG,
     1                     Y, AREA, IBD)
C
C       SUBROUTINE ASSGNBETA PROMPTS THE USER TO SPECIFY WHETHER
C       TABULATED OR CALCULATED BETA CUMULATIVE PROBABILITY DISTRIBUTIONS
C       WILL BE USED TO MODEL PARAMETER AND ALGORITHMIC UNCERTAINTIES.
C       THE CONTROL PARAMETER WHICH DICTATES WHICH OF THE TWO WILL BE
C       USED IS 'ITABL' AND IS READ BY THE SUBROUTINE.  THE ACTION
C       RESULTING FROM A SPECIFIED POSSIBLE VALUE OF ITABL IS AS FOLLOWS:
C
C         ITABL = 1: SUBROUTINE BETATABL IS CALLED IN ORDER TO ASSIGN
C                    THE APPROPRIATE TABULATED BETA CUMULATIVE PROBABILITY
C                    DISTRIBUTION TO EACH PARAMETER AND ALGORITHM WITH
C                    UNCERTAINTY.
C
C         ITABL = 2: SUBROUTINE BETACALC IS CALLED TO CALCULATE THE SPECIFIC
C                    BETA CUMULATIVE PROBABILITY DISTRIBUTION WHICH SHOULD
C                    BE ASSOCIATED WITH EACH PARAMETER AND ALGORITHM WITH
C                    UNCERTAINTY.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      REAL*4    ANOM(*), ALO(*), AHI(*), AKLO(*), AKHI(*)
      REAL*4    ASIG(2,*), Y(*), AREA(*)
      INTEGER*4 IBD(*)
C
C       INQUIRE WHETHER TABULATED OR CALCULATED BETA CUMULATIVE
C       PROBABILITY DISTRIBUTIONS ARE TO BE USED.
C
      WRITE(NOUT,1000)
 1000 FORMAT(/,' SELECT WHETHER TO USE TABULATED OR CALCULATED',/,
     1         ' BETA CUMULATIVE PROBABILITY DISTRIBUTION VALUES.',/,
     2         '         1 - USE TABULATED BETAS.',/,
     3         '         2 - USE CALCULATED BETAS.',/)
      READ(NINP,*) ITABL
C
C       FOR ITABL = 1, DIVERT TO SUBROUTINE BETATABL TO ASSIGN THE
C       APPROPRIATE TABULATED BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C
      IF (ITABL.EQ.1) THEN
        CALL BETATABL(ANOM, AKLO, AKHI, IBD)
C
C       FOR ITABL = 2, DIVERT TO SUBROUTINE BETACALC TO CALCULATE
C       THE REQUIRED BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C
```

```
ELSE IF (ITABL.EQ.2) THEN
  CALL BETACALC(ANOM, ALO, AHI, ASIG, Y, AREA, IBD)
ENDIF
RETURN
END
```

```
      SUBROUTINE BETATABL(ANOM, AKLO, AKHI, IBD)
C
C      SUBROUTINE BETATABL USES THE USER SPECIFIED PARAMETER HIGH-SIDE
C      AND LOW-SIDE K-FACTOR UNCERTAINTIES TO DETERMINE THE TABULATED
C      BETA CUMULATIVE PROBABILITY DISTRIBUTION WHICH WILL BE USED TO
C      MODEL THE UNCERTAINTY ASSOCIATED WITH EACH PARAMETER.  THE
C      SPECIFIED K-FACTORS FOR A PARAMETER ARE FIRST USED TO CALCULATE
C      THE 'MEAN' (TEMP) OF THE ASSOCIATED DISTRIBUTION:
C
C                  TEMP = (1. - 1./AKLO)/(AKHI - 1./AKLO)
C
C      THIS 'MEAN' IS THEN COMPARED TO THE 'MEAN' (11-TH ELEMENT) OF EACH
C      TABULATED BETA CUMULATIVE PROBABILITY DISTRIBUTION.  THE LATTER
C      WHICH HAS A 'MEAN' CLOSEST TO THAT DETERMINED WITH THE SPECIFIED
C      PARAMETER K-FACTORS IS THEN USED TO MODEL THE UNCERTAINTY
C      ASSOCIATED WITH THAT PARAMETER.
C
C      THE MAIN ARRAYS USED BY THE SUBROUTINE ARE:
C
C        BD   - ARRAY CONTAINING 81 TABULATED BETA CUMULATIVE PROBABILITY
C               DISTRIBUTIONS.  EACH DISTRIBUTION CONSISTS OF 21 VALUES
C               WHICH CORRESPOND TO THE FRACTION OF THE UNIT INTERVAL
C               (FROM 0 TO 1) AT WHICH 5% INCREMENTS IN THE CUMULATIVE
C               PROBABILITY ARE REACHED.  FOR INSTANCE, IF THE FIRST THREE
C               ENTRIES IN THE SET OF DATA FOR A DISTRIBUTION ARE 0.0, 0.113,
C               AND 0.187, THEN THE CUMULATIVE PROBABILITY ASSOCIATED WITH
C               THE ENTRIES IS 0.0, 0.05, AND 0.10, RESPECTIVELY.
C
C        IBD  - INTEGER ARRAY WHOSE I-TH ENTRY IS THE NUMBER OF THE TABULATED
C               BETA CUMULATIVE PROBABILITY DISTRIBUTION WHICH IS IDENTIFIED
C               AS HAVING A MEDIAN NEAREST IN VALUE TO THE COMPUTED MEDIAN
C               OF THE I-TH PARAMETER WHICH HAS BETA DISTRIBUTED UNCERTAINTY.
C               THE TABULATED DISTRIBUTION THUS IDENTIFIED IS USED TO MODEL
C               THE UNCERTAINTY ASSOCIATED WITH THE PARAMETER.
C
C        ANOM - ARRAY OF PARAMETER NOMINAL VALUES.
C        AKLO - ARRAY OF PARAMETER LOW-SIDE K-FACTOR UNCERTAINTIES.
C        AKHI - ARRAY OF PARAMETER HIGH-SIDE K-FACTOR UNCERTAINTIES.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      COMMON /BLK02/ BD(21,81)
      REAL*4    ANOM(*), AKLO(*), AKHI(*)
      REAL*4    DBA(21,41), DBB(21,40)
      INTEGER*4 IBD(*)
```

```
      DATA DBA/0.,.015,.029,.044,.059,.075,.091,.109,.127,.146,.167
     *          ,.189,.213,.239,.269,.302,.340,.386,.446,.535,1.,
     *        0.,.016,.031,.047,.062,.079,.096,.113,.132,.151,.172
     *          ,.195,.219,.245,.275,.308,.346,.392,.452,.540,1.,
     *        0.,.022,.038,.054,.070,.087,.104,.122,.141,.161,.182
     *          ,.205,.229,.255,.285,.318,.356,.402,.461,.548,1.,
     *        0.,.020,.037,.053,.070,.088,.105,.124,.143,.163,.185
     *          ,.208,.232,.259,.289,.322,.360,.406,.465,.552,1.,
     *        0.,.024,.042,.060,.077,.095,.113,.132,.151,.172,.194
     *          ,.217,.241,.268,.298,.331,.369,.415,.474,.560,1.,
     *        0.,.024,.044,.062,.081,.099,.118,.137,.157,.178,.200
     *          ,.224,.249,.276,.305,.339,.377,.423,.481,.566,1.,
     *        0.,.029,.050,.069,.088,.107,.126,.146,.166,.188,.210
     *          ,.233,.259,.286,.316,.349,.387,.432,.490,.575,1.,
     *        0.,.030,.052,.072,.092,.112,.132,.152,.173,.194,.217
     *          ,.241,.266,.293,.323,.357,.395,.440,.498,.581,1.,
     *        0.,.035,.058,.080,.100,.120,.141,.162,.183,.205,.227
     *          ,.251,.277,.304,.334,.368,.406,.450,.507,.590,1.,
     *        0.,.039,.064,.087,.108,.129,.150,.171,.193,.215,.238
     *          ,.263,.288,.316,.346,.379,.417,.461,.518,.599,1.,
     *        0.,.043,.070,.094,.117,.138,.160,.182,.204,.227,.250
     *          ,.274,.300,.328,.358,.391,.429,.473,.529,.609,1.,
     *        0.,.046,.074,.098,.121,.143,.165,.187,.210,.232,.256
     *          ,.281,.306,.334,.364,.397,.435,.479,.534,.614,1.,
     *        0.,.050,.078,.103,.127,.150,.172,.194,.217,.240,.263
     *          ,.288,.314,.342,.372,.405,.442,.486,.541,.620,1.,
     *        0.,.053,.083,.109,.133,.156,.178,.201,.224,.247,.271
     *          ,.295,.321,.349,.379,.412,.449,.493,.547,.625,1.,
     *        0.,.057,.087,.114,.138,.162,.185,.208,.230,.254,.278
     *          ,.303,.329,.356,.386,.419,.456,.499,.553,.631,1.,
     *        0.,.061,.093,.120,.145,.169,.192,.215,.239,.262,.286
     *          ,.311,.337,.365,.395,.427,.464,.507,.560,.637,1.,
     *        0.,.065,.098,.126,.152,.176,.200,.223,.246,.270,.294
     *          ,.319,.345,.373,.403,.435,.472,.514,.567,.643,1.,
     *        0.,.070,.105,.134,.160,.184,.208,.232,.255,.279,.303
     *          ,.329,.355,.382,.412,.444,.481,.523,.575,.650,1.,
     *        0.,.076,.111,.141,.168,.193,.217,.241,.264,.288,.313
     *          ,.338,.364,.391,.421,.453,.489,.531,.583,.656,1.,
     *        0.,.082,.119,.149,.176,.202,.226,.250,.274,.298,.323
     *          ,.348,.374,.401,.431,.463,.498,.540,.591,.664,1.,
     *        0.,.089,.127,.158,.186,.212,.237,.261,.285,.309,.334
     *          ,.359,.385,.412,.441,.473,.508,.549,.600,.671,1.,
     *        0.,.092,.131,.162,.190,.216,.241,.266,.290,.314,.338
     *          ,.363,.389,.417,.446,.478,.513,.554,.604,.675,1.,
     *        0.,.096,.136,.168,.196,.223,.248,.273,.297,.321,.345
     *          ,.370,.396,.424,.453,.484,.519,.560,.609,.680,1.,
```

```
*        0.,.100,.141,.173,.202,.228,.254,.278,.302,.327,.351
*          ,.376,.402,.429,.458,.489,.524,.565,.614,.683,1.,
*        0.,.104,.145,.178,.207,.234,.259,.284,.308,.332,.357
*          ,.382,.408,.435,.464,.495,.529,.569,.618,.687,1.,
*        0.,.109,.151,.184,.213,.240,.266,.290,.315,.339,.363
*          ,.388,.414,.441,.470,.501,.535,.575,.624,.692,1.,
*        0.,.113,.156,.190,.219,.246,.272,.297,.321,.346,.370
*          ,.395,.421,.448,.476,.507,.541,.580,.629,.696,1.,
*        0.,.119,.162,.196,.226,.253,.279,.304,.329,.353,.377
*          ,.402,.428,.455,.483,.514,.548,.587,.634,.701,1.,
*        0.,.124,.168,.203,.233,.260,.286,.311,.336,.360,.385
*          ,.410,.435,.462,.490,.520,.554,.593,.640,.706,1.,
*        0.,.129,.174,.209,.240,.267,.294,.319,.343,.367,.392
*          ,.417,.442,.469,.497,.527,.560,.598,.645,.710,1.,
*        0.,.136,.181,.217,.247,.275,.301,.327,.351,.375,.400
*          ,.424,.450,.476,.504,.534,.567,.605,.651,.715,1.,
*        0.,.143,.189,.225,.256,.284,.310,.335,.360,.384,.408
*          ,.433,.458,.484,.512,.542,.574,.612,.657,.721,1.,
*        0.,.149,.197,.233,.264,.292,.319,.344,.368,.393,.417
*          ,.441,.466,.492,.520,.549,.582,.619,.663,.726,1.,
*        0.,.157,.205,.242,.273,.301,.328,.353,.378,.402,.426
*          ,.450,.475,.501,.528,.557,.589,.626,.670,.731,1.,
*        0.,.164,.213,.250,.282,.310,.337,.362,.387,.411,.435
*          ,.459,.484,.509,.536,.565,.597,.633,.676,.737,1.,
*        0.,.173,.222,.260,.291,.320,.347,.372,.396,.420,.444
*          ,.468,.493,.518,.545,.573,.604,.640,.683,.742,1.,
*        0.,.182,.233,.270,.302,.331,.358,.383,.407,.431,.455
*          ,.479,.503,.528,.554,.582,.613,.648,.691,.749,1.,
*        0.,.192,.243,.281,.313,.342,.368,.394,.418,.442,.465
*          ,.489,.513,.538,.564,.591,.622,.656,.698,.755,1.,
*        0.,.202,.254,.292,.324,.353,.380,.405,.429,.452,.476
*          ,.499,.523,.548,.573,.601,.630,.664,.705,.761,1.,
*        0.,.214,.266,.305,.337,.366,.393,.418,.442,.465,.488
*          ,.511,.535,.559,.584,.611,.640,.674,.713,.768,1.,
*        0.,.225,.279,.318,.350,.379,.405,.430,.454,.477,.500
*          ,.523,.546,.570,.595,.621,.650,.682,.721,.775,1./
 DATA DBB/0.,.232,.287,.326,.360,.389,.416,.441,.465,.489,.512
*          ,.535,.558,.582,.607,.634,.663,.695,.734,.786,1.,
*        0.,.239,.295,.336,.370,.399,.427,.452,.477,.501,.524
*          ,.548,.571,.595,.620,.647,.676,.708,.746,.798,1.,
*        0.,.245,.302,.344,.378,.409,.436,.462,.487,.511,.535
*          ,.558,.582,.606,.632,.658,.687,.719,.757,.808,1.,
*        0.,.251,.309,.352,.387,.418,.446,.472,.497,.521,.545
*          ,.569,.593,.617,.642,.669,.698,.730,.767,.818,1.,
*        0.,.258,.317,.360,.396,.427,.455,.482,.507,.532,.556
*          ,.580,.604,.628,.653,.680,.709,.740,.778,.827,1.,
```

```
*        0.,.263,.324,.367,.403,.435,.464,.491,.516,.541,.565
*          ,.589,.613,.638,.663,.690,.718,.750,.787,.836,1.,
*        0.,.269,.330,.374,.411,.443,.472,.499,.525,.550,.574
*          ,.598,.622,.647,.672,.699,.727,.758,.795,.843,1.,
*        0.,.274,.337,.381,.418,.451,.480,.508,.534,.559,.583
*          ,.607,.632,.656,.681,.708,.736,.767,.803,.851,1.,
*        0.,.279,.343,.388,.426,.458,.488,.516,.542,.567,.592
*          ,.616,.640,.665,.690,.716,.744,.775,.811,.857,1.,
*        0.,.285,.349,.395,.433,.466,.496,.524,.550,.576,.600
*          ,.625,.649,.673,.699,.725,.753,.783,.819,.864,1.,
*        0.,.290,.355,.402,.440,.473,.503,.531,.558,.583,.608
*          ,.633,.657,.681,.706,.733,.760,.791,.826,.871,1.,
*        0.,.294,.360,.407,.446,.480,.510,.538,.565,.590,.615
*          ,.640,.664,.689,.714,.740,.767,.797,.832,.876,1.,
*        0.,.299,.366,.413,.452,.486,.517,.545,.572,.598,.623
*          ,.647,.671,.696,.721,.747,.774,.804,.838,.881,1.,
*        0.,.304,.371,.420,.459,.493,.524,.552,.579,.605,.630
*          ,.654,.679,.703,.728,.754,.781,.810,.844,.887,1.,
*        0.,.308,.376,.425,.465,.499,.530,.559,.586,.612,.637
*          ,.661,.685,.710,.734,.760,.787,.816,.849,.891,1.,
*        0.,.313,.382,.431,.471,.505,.536,.565,.592,.618,.643
*          ,.668,.692,.716,.741,.766,.793,.822,.855,.896,1.,
*        0.,.317,.386,.435,.476,.511,.542,.571,.598,.624,.649
*          ,.673,.698,.722,.746,.772,.798,.827,.859,.900,1.,
*        0.,.320,.391,.440,.481,.516,.547,.576,.604,.630,.655
*          ,.679,.703,.727,.752,.777,.804,.832,.864,.904,1.,
*        0.,.325,.396,.446,.487,.522,.554,.583,.611,.637,.662
*          ,.686,.710,.734,.759,.784,.810,.838,.869,.908,1.,
*        0.,.329,.400,.451,.492,.527,.559,.588,.615,.641,.666
*          ,.691,.715,.739,.763,.788,.814,.842,.873,.911,1.,
*        0.,.336,.409,.460,.502,.537,.569,.599,.626,.652,.677
*          ,.702,.726,.750,.774,.798,.824,.851,.881,.918,1.,
*        0.,.344,.417,.469,.511,.547,.579,.609,.636,.662,.687
*          ,.712,.736,.759,.783,.807,.832,.859,.889,.924,1.,
*        0.,.350,.425,.477,.519,.556,.588,.618,.645,.671,.697
*          ,.721,.745,.768,.792,.816,.840,.866,.895,.930,1.,
*        0.,.357,.433,.486,.528,.565,.597,.627,.655,.681,.706
*          ,.730,.754,.777,.800,.824,.848,.874,.902,.935,1.,
*        0.,.363,.440,.493,.536,.573,.605,.635,.663,.689,.714
*          ,.738,.761,.785,.808,.831,.855,.880,.907,.939,1.,
*        0.,.369,.447,.501,.544,.581,.614,.644,.671,.697,.722
*          ,.746,.770,.792,.815,.838,.862,.886,.913,.943,1.,
*        0.,.375,.453,.507,.551,.588,.621,.651,.679,.705,.729
*          ,.753,.776,.799,.822,.844,.867,.891,.917,.947,1.,
*        0.,.380,.459,.514,.558,.595,.628,.658,.686,.712,.737
*          ,.760,.783,.806,.828,.850,.873,.897,.922,.950,1.,
```

```
     *        0.,.386,.466,.521,.565,.603,.636,.666,.694,.719,.744
     *          ,.768,.790,.813,.835,.857,.879,.902,.926,.954,1.,
     *        0.,.391,.471,.527,.571,.609,.642,.672,.700,.726,.750
     *          ,.773,.796,.818,.840,.862,.883,.906,.930,.957,1.,
     *        0.,.401,.483,.539,.584,.622,.655,.685,.712,.738,.762
     *          ,.785,.808,.829,.851,.872,.893,.914,.937,.962,1.,
     *        0.,.410,.493,.550,.595,.633,.666,.696,.724,.749,.773
     *          ,.796,.818,.839,.860,.881,.901,.922,.943,.966,1.,
     *        0.,.419,.503,.560,.605,.643,.677,.707,.734,.759,.783
     *          ,.806,.827,.848,.868,.888,.908,.928,.948,.970,1.,
     *        0.,.426,.511,.568,.614,.652,.685,.715,.743,.768,.791
     *          ,.814,.835,.855,.875,.895,.914,.933,.952,.973,1.,
     *        0.,.434,.519,.577,.623,.661,.695,.724,.751,.776,.800
     *          ,.822,.843,.863,.882,.901,.920,.938,.956,.976,1.,
     *        0.,.441,.528,.586,.632,.671,.704,.734,.761,.785,.808
     *          ,.830,.851,.870,.889,.908,.925,.943,.960,.979,1.,
     *        0.,.448,.535,.594,.640,.678,.712,.741,.768,.793,.815
     *          ,.837,.857,.876,.895,.913,.930,.947,.964,.981,1.,
     *        0.,.453,.541,.600,.646,.684,.718,.747,.774,.798,.821
     *          ,.842,.862,.881,.899,.917,.933,.950,.966,.982,1.,
     *        0.,.460,.548,.608,.654,.692,.726,.755,.781,.806,.828
     *          ,.849,.869,.887,.905,.922,.938,.954,.969,.984,1.,
     *        0.,.465,.554,.614,.660,.699,.732,.761,.787,.811,.833
     *          ,.854,.873,.892,.909,.925,.941,.956,.971,.985,1./
      EQUIVALENCE (BD(1,1),DBA(1,1))
      EQUIVALENCE (BD(1,42),DBB(1,1))
      BDMAX=20.0
C
C      LOOP OVER THE PARAMETERS WITH UNCERTAINTY TO BE MODELED WITH
C      BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.
C
      DO 20 I = 1, IPARAM
C
C      EVALUATE THE 'MEAN' = TEMP OF THE PARAMETER AS DETERMINED FROM
C      ITS SPECIFIED K-FACTOR UNCERTAINTIES.
C
        TEMP = (1.0-1.0/AKLO(I))/(AKHI(I)-1.0/AKLO(I))
C
C      LOOPING OVER ALL THE TABULATED BETA CUMULATIVE DISTRIBUTIONS,
C      COMPARE THE PARAMETER COMPUTED 'MEAN' TO THOSE (TEMP1 AND TEMP2)
C      OF THE TABULATED BETA CUMULATIVE PROBABILITY DISTRIBUTIONS.  THE
C      'MEAN' OF THE J-TH TABULATED DISTRIBUTION IS THE 11-TH ELEMENT IN
C      THE ARRAY BD(11,J).  IF THE J-TH TABULATED DISTRIBUTION HAS A
C      'MEAN' CLOSEST TO THE COMPUTED 'MEAN' OF THE I-TH PARAMETER, ASSIGN
C      THAT DISTRIBUTION TO THE I-TH PARAMETER (IBD(I) = J).
C
```

```
            TEMP1 = 0.5*(BD(11,1)+BD(11,2))
            TEMP2 = 0.5*(BD(11,80)+BD(11,81))
            IF (TEMP.LE.TEMP1) THEN
               IBD(I) = 1
            ELSE IF (TEMP.GT.TEMP2) THEN
               IBD(I) = 81
            ELSE
               DO 10 J = 2, 80
                  TEMP1 = 0.5*(BD(11,J)+BD(11,J-1))
                  TEMP2 = 0.5*(BD(11,J)+BD(11,J+1))
                  IF ((TEMP.GT.TEMP1).AND.(TEMP.LE.TEMP2)) IBD(I) = J
   10          CONTINUE
            ENDIF
   20    CONTINUE
C
C        OUTPUT THE NUMBER OF THE CUMULATIVE BETA PROBABILITY
C        DISTRIBUTION ASSIGNED TO EACH PARAMETER WITH UNCERTAINTY.
C
         WRITE(NPRT,1000)
         WRITE(NOUT,1000)
         DO 30 I = 1, IPARAM
            WRITE(NPRT,1100) I, ANOM(I), AKLO(I), AKHI(I), IBD(I)
            WRITE(NOUT,1100) I, ANOM(I), AKLO(I), AKHI(I), IBD(I)
   30    CONTINUE
         RETURN
 1000    FORMAT(/,' SELECTED TABULATED BETA CUMULATIVE PDF ',/,
        1          '  PARAMETER    NOMINAL   UNCERTAINTY FACTORS   BETA ',/,
        2          '   NUMBER       VALUE     K-LOW      K-HIGH   NUMBER',/)
 1100    FORMAT(3X,I5,2X,1PE10.4,2X,0PF10.3,2X,0PF10.3,5X,I5)
         END
```

```
      SUBROUTINE BETACALC(ANOM, ALO, AHI, ASIG, Y, AREA, IBD)
C
C      SUBROUTINE BETACALC IS USED TO COMPUTE THE BETA CUMULATIVE
C      PROBABILITY DISTRIBUTION TO BE USED TO MODEL THE UNCERTAINTY
C      ASSOCIATED WITH EACH PARAMETER WITH BETA DISTRIBUTED UNCERTAINTY.
C      FIRST THE NOMINAL, LOW, AND HIGH VALUES OF THE PARAMETER ARE
C      USED TO COMPUTE THE 'MEAN' M OF THE DESIRED DISTRIBUTION:
C
C                    M = (ANOM - ALO)/(AHI - ALO)
C
C      THIS 'MEAN' IS THEN USED TO COMPUTE THE POSSIBLE MAXIMUM
C      STANDARD DEVIATIONS (SIG1 AND SIG2) FOR THE PARAMETER:
C
C                   SIG1 = M*SQRT((1 - M)/(1 + M))
C                   SIG2 = (1 - M)*SQRT(M/(2 - M))
C
C      THE USER IS THEN REQUESTED TO SPECIFY THE STANDARD DEVIATION SIG
C      FOR THE PARAMETER WHICH MUST SATISFY THE FOLLOWING:
C
C              SIG LESS THAN OR EQUAL TO MIN(SIG1, SIG2)
C
C      THE SPECIFIED STANDARD DEVIATION FOR THE PARAMETER IS NEXT USED
C      TO COMPUTE A 'NORMALIZED' STANDARD DEVIATION SIGMA:
C
C                       SIGMA = SIG/(AHI - ALO)
C
C      AND THE TWO PARAMETERS OF THE BETA DISTRIBUTION (ALPHA AND BETA)
C      ARE THEN COMPUTED:
C
C                  ALPHA = M*[{(M - M*M)/SIGMA**2} - 1]
C                  BETA  = (1 - M)*[{(M - M*M)/SIGMA**2} - 1]
C
C      FUNCTION ROUTINE FX IS NEXT USED TO EVALUATE THE BETA DISTRIBUTION
C      CHARACTERIZED BY THE PARAMETERS ALPHA AND BETA, AND THE ASSOCIATED
C      CUMULATIVE PROBABILITY DISTRIBUTION (AT 5% AREA INCREMENTS) IS
C      DETERMINED AND SAVED FOR USE IN MODELING THE PARAMETER UNCERTAINTY.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      COMMON /BLK02/ BD(21,81)
      REAL*4    ANOM(*), ALO(*), AHI(*)
      REAL*4    ASIG(2,*), Y(*), AREA(*)
      INTEGER*4 IBD(*)
C
C      DEFINE THE INCREMENT USED IN EVALUATING THE BETA DISTRIBUTION AND
```

```
C      CALCULATING THE AREA INCREMENTS UNDER THE DISTRIBUTION.
C
       DELINC = 1.0/NINC
C
C      COMPUTE THE MAXIMUM ALLOWABLE STANDARD DEVIATION FOR
C      THE PARAMETER.  LOOP OVER ALL PARAMETERS WITH UNCERTAINTY.
C
       DO 20 I = 1, IPARAM
          TEM = (ANOM(I) - ALO(I))/(AHI(I) - ALO(I))
          ASIG(1,I) = TEM*SQRT((1.0 - TEM)/(1.0 + TEM))
          ASIG(1,I) = (AHI(I) - ALO(I))*ASIG(1,I)
          ASIG(2,I) = (1.0 - TEM)*SQRT(TEM/(2.0 - TEM))
          ASIG(2,I) = (AHI(I) - ALO(I))*ASIG(2,I)
    20 CONTINUE
       DO 30 I = 1, IPARAM
          WRITE(NPRT,1000) I, ASIG(1,I), ASIG(2,I)
          WRITE(NOUT,1000) I, ASIG(1,I), ASIG(2,I)
    30 CONTINUE
C
C      INTERACTIVELY READ THE PARAMETER SPECIFIED STANDARD DEVIATION.
C
       WRITE(NPRT,1100)
       WRITE(NOUT,1100)
       READ(NINP,*) (ASIG(1,I), I = 1, IPARAM)
C
C      ECHO THE INPUT VALUES.
C
       WRITE(NPRT,1200)
       WRITE(NOUT,1200)
       DO 40 I = 1, IPARAM
          WRITE(NPRT,1300) I, ANOM(I), ALO(I), AHI(I), ASIG(1,I)
          WRITE(NOUT,1300) I, ANOM(I), ALO(I), AHI(I), ASIG(1,I)
    40 CONTINUE
       WRITE(NPRT,1400)
       WRITE(NOUT,1400)
C
C      LOOP OVER PARAMETERS WITH UNCERTAINTY.
C
       DO 90 I = 1, IPARAM
C
C      COMPUTE THE NORMALIZED PARAMETER STANDARD DEVIATION, PARAMETER
C      NORMALIZED MEAN, AND THE ALPHA AND BETA PARAMETERS FOR THE
C      APPROPRIATE BETA DISTRIBUTION.
C
          ASIG(1,I) = ASIG(1,I)/(AHI(I) - ALO(I))
          TEM = (ANOM(I) - ALO(I))/(AHI(I) - ALO(I))
```

```
          TEM1 = (TEM - TEM*TEM)/(ASIG(1,I)*ASIG(1,I)) - 1.0
          ALPHA = TEM*TEM1
          BETA  = (1.0 - TEM)*TEM1
C
C       OUTPUT THE COMPUTED ALPHA AND BETA PARAMETERS.
C
          WRITE(NPRT,1500) I, ALPHA, BETA
          WRITE(NOUT,1500) I, ALPHA, BETA
C
C       COMPUTE THE BETA DISTRIBUTION WITH PARAMETER VALUES
C       ALPHA AND BETA.  USE THE TRAPEZOIDAL RULE TO COMPUTE
C       THE TOTAL AREA (SUMNORM) UNDER THE COMPUTED BETA
C       DISTRIBUTION.
C
          Y(1) = 0.0
          SUMNORM = 0.0
          DO 50 J = 1, NINC
             Y(J+1) = Y(J) + DELINC
             FX1 = FX(ALPHA, BETA, Y(J))
             FX2 = FX(ALPHA, BETA, Y(J+1))
             SUMNORM = SUMNORM + 0.5*(FX1 + FX2)*(Y(J+1) - Y(J))
   50     CONTINUE
C
C       COMPUTE THE NORMALIZED BETA DISTRIBUTION.
C
          DO 60 J = 1, NINC+1
             FX1 = FX(ALPHA, BETA, Y(J))
             Y(J) = FX1/SUMNORM
   60     CONTINUE
C
C       EVALUATE AREA INCREMENTS FOR THE AREA UNDER THE NORMALIZED
C       BETA DISTRIBUTION.
C
          AREA(1) = 0.5*DELINC*(Y(1) + Y(2))
          DO 70 J = 2, NINC
             AREA(J) = AREA(J-1) + 0.5*DELINC*(Y(J) + Y(J+1))
   70     CONTINUE
C
C       DETERMINE THE ABSCISSA VALUES FOR THE 5% AREA INCREMENTS
C       (HENCE THE CUMULATIVE PROBABILITY DISTRIBUTION) FOR THE
C       COMPUTED NORMALIZED BETA DISTRIBUTION.  SAVE THE COMPUTED
C       VALUES IN THE ARRAY BD(J,I) (J = 1, 21) FOR THE I-TH
C       PARAMETER.
C
          BD(1,I)  = 0.0
          BD(21,I) = 1.0
```

```
         IBD(I)   = I
         DO 90 J = 2, 20
            AAREA = 0.05*(J - 1)
            DO 80 K = 1, NINC-1
            IF (AAREA.GT.AREA(K+1)) THEN
               GO TO 80
            ELSE IF ((AAREA.GT.AREA(K)).AND.(AAREA.LE.AREA(K+1))) THEN
               FRAC = (AAREA - AREA(K))/(AREA(K+1) - AREA(K))
               BD(J,I) = K*DELINC + FRAC*DELINC
               GO TO 90
            ELSE IF (AAREA.LE.AREA(K)) THEN
               GO TO 90
            ENDIF
   80       CONTINUE
   90 CONTINUE
C
C      OUTPUT THE BETA CUMULATIVE PROBABILITY DISTRIBUTION COMPUTED
C      FOR EACH PARAMETER WITH UNCERTAINTY.
C
      DO 110 I = 1, IPARAM
         WRITE(NPRT,1600) I
         WRITE(NOUT,1600) I
         DO 100 J = 1, 10
            FRAC1 = (J - 1)*0.05
            FRAC2 = (J + 10)*0.05
            WRITE(NPRT,1700) FRAC1, BD(J,I), FRAC2, BD(J+11,I)
            WRITE(NOUT,1700) FRAC1, BD(J,I), FRAC2, BD(J+11,I)
  100    CONTINUE
         WRITE(NPRT,1700) 0.50, BD(11,I)
         WRITE(NOUT,1700) 0.50, BD(11,I)
  110 CONTINUE
      RETURN
 1000 FORMAT(' FOR PARAMETER',I3,' THE STAN. DEV. MUST BE <',
     1       ' MIN(',1PE10.4,',',1PE10.4,').')
 1100 FORMAT(/,' ENTER THE STANDARD DEVIATIONS OF THE BETA',/,
     1         ' DISTRIBUTED PARAMETERS.',/)
 1200 FORMAT(/,' PARAMETER    NOMINAL    MINIMUM   MAXIMUM   ',
     1         ' STAN. DEV.',/)
 1300 FORMAT(I5,5X,1PE10.4,2X,1PE10.4,2X,1PE10.4,2X,1PE10.4)
 1400 FORMAT(/,' PARAMETER     ALPHA      BETA',/)
 1500 FORMAT(3X,I5,5X,F7.3,5X,F7.3)
 1600 FORMAT(/,' FOR PARAMETER',I3,' THE CUMULATIVE BETA',/,
     1         ' DISTRIBUTION VALUES AT 5% AREA INCREMENTS',/,
     2         ' ARE AS FOLLOWS:',/,/,
     3         '     AREA                AREA',/,
     4         '   FRACTION    VALUE   FRACTION   VALUE',/)
```

```
1700 FORMAT(5X,F5.2,5X,F7.3,5X,F5.2,5X,F7.3)
     END
```

```
      SUBROUTINE SOLVE(ANOM, ALO, AHI, AKLO, AKHI, XVAL, A, IBD, RESULT)
C
C     SUBROUTINE SOLVE EVALUATES THE USER ALGORITHM ISIZE TIMES IN
C     ORDER TO GENERATE A ALGORITHM RESULTS DISTRIBUTION FROM WHICH
C     THE OVERALL COMBINED UNCERTAINTY TO BE ASSOCIATED WITH THE OUTPUT
C     OF THE ALGORITHM IS DETERMINED.  THE ALGORITHM IS EVALUATED IN A
C     TWO STEP PROCESS:
C
C       1. THE PARAMETER VALUES TO BE USED IN EVALUATION OF THE ALGORITHM
C          ARE RANDOMIZED.  A NEW RANDOM NUMBER (RN) IS USED EACH TIME
C          TO GENERATE A NEW RANDOMIZED VALUE FOR EACH PARAMETER WITH
C          UNCERTAINTY IN THE ALGORITHM.  EACH RANDOM NUMBER USED FOR
C          OBTAINING VALUES FOR PARAMETERS WITH NORMALLY OR LOGNORMALLY
C          DISTRIBUTED UNCERTAINTY IS NORMALLY DISTRIBUTED AND OBTAINED
C          BY CALLING SUBROUTINE FNRN.  THE RANDOM NUMBERS USED FOR
C          PARAMETERS WHICH ARE BETA OR UNIFORMLY DISTRIBUTED ARE OBTAINED
C          BY CALLING SUBROUTINE UPR1 AND ARE UNIFORMLY DISTRIBUTED.  A RANDO
C          VALUE FOR A PARAMETER IS A FUNCTION OF THE PARAMETER NOMINAL
C          VALUE, THE UNCERTAINTY SPECIFICATION, AND THE CURRENT RANDOM
C          NUMBER VALUE.  IF NECESSARY, NOMINAL VALUES OF PARAMETERS
C          WHICH ARE GENERATED FROM SIMPLE ALGORITHMS ARE FIRST OBTAINED
C          BY CALLING SUBROUTINE AUXALGO.  THIS WILL ONLY BE DONE IF IXALG =
C
C       2. THE ALGORITHM, WHICH IS CODED IN THE USER SUPPLIED SUBROUTINE TF,
C          IS THEN EVALUATED USING THE CURRENT SET OF RANDOMIZED PARAMETER
C          VALUES.  THE RESULT OF THE EVALUATION IS RETURNED TO SUBROUTINE
C          SOLVE AND, IF NECESSARY (IALGO = 1), ALGORITHMIC UNCERTAINTY IS
C          APPLIED TO THE RESULT.  THE FINAL COMPUTED VALUE FOR THE ALGORITHM
C          IS THEN STORED IN ARRAY RESULT.
C
C     THE ABOVE TWO STEP PROCESS IS REPEATED THE USER SPECIFIED ISIZE TIMES.
C
C     THE MAIN VARIABLES AND ARRAYS USED BY THE SUBROUTINE ARE AS FOLLOWS:
C
C       ANOM   - ARRAY OF NOMINAL VALUES OF PARAMETERS WITH UNCERTAINTY.
C       AKLO   - ARRAY OF PARAMETER LOW-SIDE K-FACTOR UNCERTAINTIES.
C       AKHI   - ARRAY OF PARAMETER HIGH-SIDE K-FACTOR UNCERTAINTIES.
C       ALO    - ARRAY OF PARAMETER COMPUTED LOW-SIDE EXTREME VALUES.
C       AHI    - ARRAY OF PARAMETER COMPUTED HIGH-SIDE EXTREME VALUES.
C       A      - TEMPORARY ARRAY OF RANDOMIZED PARAMETER VALUES.
C       XVAL   - ARRAY OF INDEPENDENT PARAMETER VALUES (NO UNCERTAINTY).
C       BD     - ARRAY OF TABULATED OR CALCULATED BETA CUMULATIVE
C                PROBABILITY DISTRIBUTIONS.
C       IBD    - ARRAY WHOSE ENTRIES IDENTIFY THE DISTRIBUTION NUMBER
C                ASSOCIATED WITH EACH PARAMETER WITH BETA DISTRIBUTED
```

```
C                 UNCERTAINTY.
C        RESULT - ARRAY OF EVALUATED ALGORITHM RESULTS.
C        RN     - VARIABLE WHOSE VALUE IS THE CURRENT NORMALLY OR
C                 UNIFORMLY DISTRIBUTED RANDOM NUMBER.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      COMMON /BLK02/ BD(21,81)
      REAL*4    ANOM(*), ALO(*), AHI(*), AKLO(*), AKHI(*)
      REAL*4    XVAL(*), A(*), RESULT(*)
      INTEGER*4 IBD(*)
      BDMAX = 20.0
C
C      INITIALIZE THE SEED INTEGERS FOR THE RANDOM NUMBER GENERATORS.
C
      NUMUR = ISEED
      NUMNR = ISEED
C
C      SET MAXIMUM LOOP COUNTER VALUE IPARA DEPENDING ON WHETHER OR NOT
C      THE ALGORITHM TO BE EVALUATED HAS OVERALL ALGORITHMIC UNCERTAINTY.
C
      IF (IALGO.EQ.1) THEN
         IPARA = IPARAM - 1
      ELSE
         IPARA = IPARAM
      ENDIF
C
C      OBTAIN PARAMETER NOMINAL VALUES FROM SUBROUTINE AUXALGO IF REQUIRED
C      AND THEN COMPUTE ASSOCIATED LOW-SIDE AND HIGH-SIDE VALUES.
C
      IF (IXALG.EQ.1) CALL AUXALGO(ANOM, XVAL)
      IF (ITYPE.EQ.1) THEN
         DO 10 I = 1, IPARA
            AHI(I) = ANOM(I)*AKLO(I)
            ALO(I) = ANOM(I) - (AHI(I) - ANOM(I))
   10    CONTINUE
      ELSE IF (ITYPE.EQ.2) THEN
         DO 20 I = 1, IPARA
            ALO(I) = ANOM(I)/AKLO(I)
            AHI(I) = ANOM(I)*AKLO(I)
   20    CONTINUE
      ELSE IF (ITYPE.EQ.3) THEN
         DO 30 I = 1, IPARA
            ALO(I) = ANOM(I)/AKLO(I)
            AHI(I) = ANOM(I)*AKHI(I)
   30    CONTINUE
```

```
      ELSE IF (ITYPE.EQ.4) THEN
         DO 40 I = 1, IPARA
            ALO(I) = ANOM(I)/AKLO(I)
            AHI(I) = ANOM(I)*AKHI(I)
   40    CONTINUE
      ENDIF
C
C     ALGORITHM EVALUATION FOR CASE OF NORMALLY DISTRIBUTED UNCERTAINTY.
C
      IF (ITYPE.EQ.1) THEN
C
C     EVALUATE THE ALGORITHM THE USER REQUESTED (ISIZE) NUMBER OF TIMES.
C
       DO 60 I = 1, ISIZE
C
C     RANDOMIZE THE PARAMETER VALUES.
C
          DO 50 J = 1, IPARA
             CALL FNRN(NUMNR,RN)
             A(J) = ANOM(J) + RN*(AHI(J) - ANOM(J))/1.96
   50     CONTINUE
C
C     EVALUATE THE ALGORITHM USING THE RANDOMIZED PARAMETER VALUES.
C
          CALL TF(ANOM,A,XVAL,YVAL)
C
C     APPLY ALGORITHMIC UNCERTAINTY IF REQUIRED (IALGO = 1)
C     AND STORE THE RESULT OF THE ALGORITHM EVALUATION.
C
          IF (IALGO.NE.1) THEN
             RESULT(I) = YVAL
          ELSE
             CALL FNRN(NUMNR,RN)
             AHI(IPARAM) = YVAL*AKLO(IPARAM)
             RESULT(I) = YVAL + RN*(AHI(IPARAM) - YVAL)/1.96
          ENDIF
   60    CONTINUE
         RETURN
C
C     ALGORITHM EVALUATION FOR CASE OF LOGNORMALLY DISTRIBUTED UNCERTAINTY.
C
      ELSE IF (ITYPE.EQ.2) THEN
C
C     EVALUATE THE ALGORITHM THE USER REQUESTED (ISIZE) NUMBER OF TIMES.
C
       DO 120 I = 1, ISIZE
```

```
C
C       RANDOMIZE THE PARAMETER VALUES.
C
            DO 110 J = 1, IPARA
               CALL FNRN(NUMNR,RN)
               A(J) = ANOM(J)*AKLO(J)**(RN/1.96)
  110       CONTINUE
C
C       EVALUATE THE ALGORITHM USING THE RANDOMIZED PARAMETER VALUES.
C
            CALL TF(ANOM,A,XVAL,YVAL)
C
C       APPLY ALGORITHMIC UNCERTAINTY IF REQUIRED (IALGO = 1)
C       AND STORE THE RESULT OF THE ALGORITHM EVALUATION.
C
            IF (IALGO.NE.1) THEN
               RESULT(I) = YVAL
            ELSE
               CALL FNRN(NUMNR,RN)
               RESULT(I) = YVAL*AKLO(IPARAM)**(RN/1.96)
            ENDIF
  120     CONTINUE
          RETURN
C
C       ALGORITHM EVALUATION FOR CASE OF BETA DISTRIBUTED UNCERTAINTY.
C
        ELSE IF (ITYPE.EQ.3) THEN
C
C       EVALUATE THE ALGORITHM THE USER REQUESTED (ISIZE) NUMBER OF TIMES.
C
          DO 220 I = 1, ISIZE
C
C       RANDOMIZE THE PARAMETER VALUES.
C
            DO 210 J = 1, IPARA
               CALL UPR1(NUMUR,RN)
               LL = IBD(J)
               FL1 = RN*BDMAX+1.0
               L = FL1
               LP1 = L+1
               FL2 = L
               FL1 = FL1 - FL2
               FL2 = 1.0 - FL1
               BDV = FL1*BD(LP1,LL) + FL2*BD(L,LL)
               A(J) = ALO(J) + BDV*(AHI(J)-ALO(J))
  210       CONTINUE
```

```
C
C       EVALUATE THE ALGORITHM USING THE RANDOMIZED PARAMETER VALUES.
C
            CALL TF(ANOM,A,XVAL,YVAL)
C
C       APPLY ALGORITHMIC UNCERTAINTY IF REQUIRED (IALGO = 1)
C       AND STORE THE RESULT OF THE ALGORITHM EVALUATION.
C
            IF (IALGO.NE.1) THEN
               RESULT(I) = YVAL
            ELSE
               CALL UPR1(NUMUR,RN)
               ALO(IPARAM) = YVAL/AKLO(IPARAM)
               AHI(IPARAM) = YVAL*AKHI(IPARAM)
               LL = IBD(IPARAM)
               FL1 = RN*BDMAX+1.0
               L = FL1
               LP1 = L+1
               FL2 = L
               FL1 = FL1 - FL2
               FL2 = 1.0 - FL1
               BDV = FL1*BD(LP1,LL) + FL2*BD(L,LL)
               RESULT(I) = ALO(IPARAM) + BDV*(AHI(IPARAM) - ALO(IPARAM))
            ENDIF
  220    CONTINUE
         RETURN
C
C       ALGORITHM EVALUATION FOR CASE OF UNIFORMLY DISTRIBUTED UNCERTAINTY.
C
       ELSE IF (ITYPE.EQ.4) THEN
C
C       EVALUATE THE ALGORITHM THE USER REQUESTED (ISIZE) NUMBER OF TIMES.
C
         DO 320 I = 1, ISIZE
C
C       RANDOMIZE THE PARAMETER VALUES.
C
            DO 310 J = 1, IPARA
               CALL UPR1(NUMUR,RN)
               IF (RN.LE.0.5) THEN
                  A(J) = ALO(J) + 2.0*RN*(ANOM(J) - ALO(J))
               ELSE
                  A(J) = ANOM(J) + 2.0*(RN - 0.5)*(AHI(J) - ANOM(J))
               ENDIF
  310       CONTINUE
C
```

```
C       EVALUATE THE ALGORITHM USING THE RANDOMIZED PARAMETER VALUES.
C
            CALL TF(ANOM,A,XVAL,YVAL)
C
C       APPLY ALGORITHMIC UNCERTAINTY IF REQUIRED (IALGO = 1)
C       AND STORE THE RESULT OF THE ALGORITHM EVALUATION.
C
            IF (IALGO.NE.1) THEN
               RESULT(I) = YVAL
            ELSE
               CALL UPR1(NUMUR,RN)
               ALO(IPARAM) = YVAL/AKLO(IPARAM)
               AHI(IPARAM) = YVAL*AKHI(IPARAM)
               IF (RN.LE.0.5) THEN
                 RESULT(I) = ALO(IPARAM) + 2.0*RN*(YVAL - ALO(IPARAM))
               ELSE
                 RESULT(I) = YVAL + 2.0*(RN - 0.5)*(AHI(IPARAM) - YVAL)
               ENDIF
            ENDIF
  320    CONTINUE
         RETURN
       ENDIF
       RETURN
       END
```

```
      SUBROUTINE SORT(X,INUM)
C
C      SUBROUTINE SORT SORTS THE INUM VALUES IN THE ARRAY X FROM LOWEST
C      TO HIGHEST VALUE.  IN THE PRESENT CONTEXT, THE SUBROUTINE IS USED
C      TO SORT THE ARRAY OF COMPUTED ALGORITHM RESULTS FROM LOWEST TO
C      HIGHEST VALUE.
C
      INTEGER*4 INUM
      REAL*4    X(*)
      NEND = INUM - 1
   10 IDONE = 0
      DO 20 I = 1, NEND
         IF (X(I).GT.X(I+1)) THEN
            XTEM = X(I)
            X(I) = X(I+1)
            X(I+1) = XTEM
            IDONE = 1
         ENDIF
   20 CONTINUE
      IF (IDONE.EQ.0) GO TO 30
      NEND = NEND - 1
      GO TO 10
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE STATISTIC(X, XINT, XOUT, XXLN, XXII, IFREQ)
C
C      SUBROUTINE STATISTIC CALCULATES VARIOUS STATISTICS FOR THE ORDERED
C      (FROM LOWEST TO HIGHEST VALUE) ALGORITHM RESULTS DISTRIBUTION WHICH
C      IS PASSED TO THE SUBROUTINE IN ARRAY X.  FOUR BASIC MANIPULATIONS OF
C      THE ALGORITHM RESULTS DISTRIBUTION ARE PERFORMED BY THE SUBROUTINE:
C
C      1. GENERATE HISTOGRAM DATA.
C
C         THE FULL RANGE (FROM X(1) TO X(ISIZE) WHERE ISIZE IS THE NUMBER
C         OF COMPUTED ALGORITHM RESULTS) OF THE RESULTS DISTRIBUTION IS
C         DIVIDED INTO NINTV EQUAL LENGTH INTERVALS, THE BOUNDARY VALUES
C         OF WHICH ARE CONTAINED IN THE ARRAY XINT (NINTV+1 ENTRIES).  THE
C         NUMBER OF RESULTS DISTRIBUTION VALUES OCCURRING IN EACH RANGE
C         INTERVAL IS THEN DETERMINED AND STORED IN THE INTEGER ARRAY IFREQ.
C         THE RESULT IS A HISTOGRAM FOR THE RESULTS DISTRIBUTION.
C
C      2. DETERMINE DATA MEAN (OR AVERAGE).
C
C         FOR NORMALLY, LOGNORMALLY, AND BETA DISTRIBUTED ALGORITHM RESULTS,
C         THE MIDPOINT OF THE I-TH RANGE SUBINTERVAL IS MULTIPLIED BY THE
C         NUMBER OF OCCURRENCES OF THE ALGORITHM VALUE IN THE SUBINTERVAL;
C
C                         0.5*(XINT(I) - XINT(I+1))*IFREQ(I)
C
C         AND THE RESULT IS SUMMED OVER ALL SUBINTERVALS.  THE EVALUATED
C         SUM IS THEN DIVIDED BY THE TOTAL NUMBER OF ALGORITHM RESULTS (ISIZE)
C         TO GENERATE THE APPROXIMATE MEAN OF THE RESULTS DISTRIBUTION.
C
C         FOR UNIFORMLY DISTRIBUTED ALGORITHM RESULTS, THE FRACTIONAL POSITION
C         IN THE INTERVAL CORRESPONDING TO THE ISIZE/2 RESULT IN THE ORDERED
C         RESULTS DISTRIBUTION IS DETERMINED AND REPORTED AS THE MEAN (AMEAN)
C         OF THE DISTRIBUTION.
C
C      3. ESTIMATE VARIANCE AND STANDARD DEVIATION:
C
C         THE COMPUTED MEAN OF THE DISTRIBUTION IS SUBTRACTED FROM THE MIDPOIN
C         VALUE (XAVE) OF EACH SUBINTERVAL TO DEFINE VARIABLE XDIF:
C
C               XDIF = 0.5*(XINT(I+1) + XINT(I)) - AMEAN = XAVE - AMEAN
C
C         THE VALUE OF XDIF FOR EACH SUBINTERVAL IS NEXT SQUARED, MULTIPLIED
C         BY THE NUMBER OF ALGORITHM RESULTS OCCURRING IN THE SUBINTERVAL, AND
C         THE RESULT FOR EACH SUBINTERVAL IS SUMMED OVER ALL THE INTERVALS.
C         THE COMPUTED SUM IS THEN DIVIDED BY THE TOTAL NUMBER OF VALUES (ISIZE)
```

```
C         IN THE RESULTS DISTRIBUTION, AND THE RESULT (VAR) IS INTERPRETED AS
C         AN APPROXIMATE VALUE FOR THE VARIANCE OF THE ALGORITHM RESULTS
C         DISTRIBUTION.  THE SQUARE ROOT OF VAR THEN PROVIDES THE ESTIMATE FOR
C         THE STANDARD DEVIATION (SIGMA) OF THE RESULTS DISTRIBUTION.
C
C      4. ESTIMATE THE DISTRIBUTION K-FACTORS:
C
C         NORMALLY DISTRIBUTED RESULTS:
C
C                              K = 1.0 + 1.96*SIGMA/AMEAN
C
C         LOGNORMALLY DISTRIBUTED RESULTS:
C
C                              K = EXP(1.96*SIGMA/AMEAN)
C
C         BETA AND UNIFORMLY DISTRIBUTED RESULTS:
C
C                            KLO = AMEAN/X(1)
C                            KHI = X(ISIZE)/AMEAN
C                           KAVE = 0.5*(KLO + KHI)
C
C     THE SUBROUTINE THEN INQUIRES WHETHER OR NOT PLOT FILES FOR THE
C     ALGORITHM RESULTS DISTRIBUTION AND FITS TO THE RESULTS ARE TO BE
C     GENERATED.  IF GENERATION OF PLOT FILES IS REQUESTED BY THE USER,
C     THE SUBROUTINE CALLS SUBROUTINE HISTO.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      REAL*4    X(*), XINT(*), XOUT(*), XXLN(*), XXII(*)
      INTEGER*4 IFREQ(*)
      CHARACTER*1 ANS2
C
C      COMPUTE INCREMENT FOR RESULTS SUBINTERVALS.
C
      DEL = (X(ISIZE) - X(1))/NINTV
C
C      GENERATE RESULTS SUBINTERVALS.
C
      XINT(1) = X(1)
      XINT(NINTV+1) = X(ISIZE)
      DO 10 I = 2, NINTV
         XINT(I) = XINT(I-1) + DEL
   10 CONTINUE
C
C      DETERMINE THE NUMBER OF ALGORITHM RESULTS OCCURRING IN EACH
C      RANGE SUBINTERVAL AND STORE IN ARRAY IFREQ.
```

```
C
      DO 20 I = 1, NINTV
         IFREQ(I) = 0
   20 CONTINUE
      DO 40 I = 1, ISIZE
         RES = X(I)
         DO 30 J = 1, NINTV
            IF (RES.GT.XINT(J+1)) THEN
               GO TO 30
            ELSE IF ((RES.GE.XINT(J)).AND.(RES.LT.XINT(J+1))) THEN
               IFREQ(J) = IFREQ(J) + 1
               GO TO 40
            ENDIF
   30    CONTINUE
   40 CONTINUE
      IFREQ(NINTV) = IFREQ(NINTV) + 1
C
C      COMPUTE THE DISTRIBUTION MEAN FOR NORMALLY, LOGNORMALLY, AND
C      BETA DISTRIBUTED ALGORITHM RESULTS DISTRIBUTIONS.
C
      IF ((ITYPE.EQ.1).OR.(ITYPE.EQ.2).OR.(ITYPE.EQ.3)) THEN
        AMEAN = 0.0
        DO 50 I = 1, NINTV
           AMEAN = AMEAN + 0.5*(XINT(I+1) + XINT(I))*IFREQ(I)
   50   CONTINUE
        AMEAN = AMEAN/ISIZE
        GO TO 70
C
C      COMPUTE THE DISTRIBUTION MEAN FOR UNIFORMLY DISTRIBUTED
C      ALGORITHM RESULTS DISTRIBUTIONS.
C
      ELSE IF (ITYPE.EQ.4) THEN
        IRESULT = 0
        DO 60 I = 1, NINTV
           IDIF = ISIZE/2 - IRESULT - IFREQ(I)
           IF (IDIF.GT.0) THEN
              IRESULT = IRESULT + IFREQ(I)
           ELSE IF (IDIF.LE.0) THEN
              FRAC = (ISIZE/2 - IRESULT)/IFREQ(I)
              AMEAN = XINT(I) + FRAC*(XINT(I+1) - XINT(I))
              GO TO 70
           ENDIF
   60   CONTINUE
      ENDIF
   70 VAR = 0.0
C
```

```
C      COMPUTE THE DISTRIBUTION VARIANCE FOR ALL ALGORITHM RESULTS
C      DISTRIBUTION TYPES.
C
       DO 80 I = 1, NINTV
          XAVE = 0.5*(XINT(I+1) + XINT(I))
          XDIF = XAVE - AMEAN
          VAR = VAR + XDIF*XDIF*IFREQ(I)
    80 CONTINUE
       VAR = VAR/ISIZE
C
C      COMPUTE THE STANDARD DEVIATION (SQRT(VAR)) OF THE DISTRIBUTION.
C
       SIGMA = SQRT(VAR)
C
C      OUTPUT THE COMPUTED MEAN, VARIANCE, AND STANDARD DEVIATION FOR
C      THE ALGORITHM RESULTS DISTRIBUTION.
C
       WRITE(NPRT,1000) AMEAN, VAR, SIGMA, ISIZE
       WRITE(NOUT,1000) AMEAN, VAR, SIGMA, ISIZE
C
C      COMPUTE AND OUTPUT THE K-FACTOR FOR NORMALLY DISTRIBUTED
C      ALGORITHM RESULTS.
C
       IF (ITYPE.EQ.1) THEN
         AKEST = 1.0 + 1.96*SIGMA/AMEAN
         WRITE(NPRT,1100) AKEST
         WRITE(NOUT,1100) AKEST
C
C      COMPUTE AND OUTPUT THE K-FACTOR FOR LOGNORMALLY DISTRIBUTED
C      ALGORITHM RESULTS.
C
       ELSE IF (ITYPE.EQ.2) THEN
         AKEST = EXP(1.96*SIGMA/AMEAN)
         WRITE(NPRT,1100) AKEST
         WRITE(NOUT,1100) AKEST
C
C      COMPUTE AND OUTPUT THE K-FACTORS (KLO, KHI, KAVE) FOR BETA AND
C      UNIFORMLY DISTRIBUTED ALGORITHM RESULTS.
C
       ELSE IF ((ITYPE.EQ.3).OR.(ITYPE.EQ.4)) THEN
         AKEST1 = AMEAN/X(1)
         AKEST2 = X(ISIZE)/AMEAN
         AKAVE = 0.5*(AKEST1 + AKEST2)
         WRITE(NPRT,1200) AKEST1, AKEST2, AKAVE
         WRITE(NOUT,1200) AKEST1, AKEST2, AKAVE
       ENDIF
```

```
C
C      DETERMINE IF A RESULTS DISTRIBUTION PLOTFILE AND A FIT TO THE
C      ALGORITHM RESULTS DISTRIBUTION ARE TO BE GENERATED.  IF SO,
C      CALL SUBROUTINE HISTO, ELSE EXIT SUBROUTINE.
C
      WRITE(NOUT,1300)
      READ(NINP,1031) ANS2
 1031 FORMAT(A1)
      IF ((ANS2.EQ. 'Y').OR.(ANS2.EQ. 'y')) THEN
         CALL HISTO(XINT, IFREQ, AMEAN, SIGMA, XOUT, XXLN, XXII)
      ENDIF
      RETURN
 1000 FORMAT(/,' MEAN OF DISTRIBUTION                = ',E12.5,/,
     1         ' VARIANCE OF DISTRIBUTION            = ',E12.5,/,
     2         ' STANDARD DEVIATION [SQRT(VARIANCE)] = ',E12.5,/,
     3         ' TOTAL NUMBER OF SAMPLES             = ',6X,I6)
 1100 FORMAT(/,' K-FACTOR ESTIMATE:  K = ',F7.4,/)
 1200 FORMAT(/,' K-FACTOR ESTIMATES:',/,/,
     1         '        K-LOW      = ',F7.4,/,
     2         '        K-HIGH     = ',F7.4,/,
     3         '        K-AVERAGE  = ',F7.4,/)
 1300 FORMAT(/,' DO YOU WANT PLOT FILES OF THE RESULTS',/,
     1         ' DISTRIBUTION AND FIT TO BE GENERATED? [Y/N]',/)
      END
```

```
      SUBROUTINE HISTO(XX, II, AMEAN, SIGMA, XOUT, XXLN, XXII)
C
C      SUBROUTINE HISTO GENERATES (X,Y) DATA FILES WHICH CAN BE USED
C      TO GENERATE PLOTS OF THE ALGORITHM RESULTS DISTRIBUTION,
C      COMPUTED FITS TO THE DISTRIBUTION RESULTS, AND THE CUMULATIVE
C      PROBABILITY DISTRIBUTION FOR THE RESULTS.  THE (X,Y) PLOT DATA
C      ARE WRITTEN TO DEFAULT UNIT 3.
C
C      THE BASIC DATA DELIVERED TO THE SUBROUTINE ARE AS FOLLOWS:
C
C          XX    - ARRAY OF DISTRIBUTION SUBINTERVAL BOUNDARIES.
C          II    - INTEGER ARRAY OF NUMBER OF ALGORITHM RESULTS
C                  OCCURRING IN EACH SUBINTERVAL.
C          AMEAN - COMPUTED MEAN (OR AVERAGE) OF THE DISTRIBUTION.
C          SIGMA - COMPUTED STANDARD DEVIATION OF THE DISTRIBUTION.
C
C      THE SUBROUTINE PERFORMS THE FOLLOWING OPERATIONS:
C
C      1. OUTPUT HISTOGRAM DATA.
C
C         THE MIDPOINT VALUE XOUT OF THE I-TH SUBINTERVAL IS COMPUTED
C
C                        XOUT(I) = 0.5*(XX(I) + XX(I+1))
C
C         AND THE DATA PAIRS (XOUT(I), II(I)) FOR EACH SUBINTERVAL ARE
C         OUTPUT TO THE PLOT FILE.
C
C      2. CURVE FIT TO HISTOGRAM DATA.
C
C         A. NORMALLY DISTRIBUTED ALGORITHM RESULTS:
C
C            THE SUBINTERVAL CONTAINING THE LARGEST NUMBER OF COMPUTED
C            ALGORITHM RESULTS IS DETERMINED AND INTEGER VARIABLE IIMAX
C            IS SET EQUAL TO THE NUMBER OF ALGORITHM RESULTS WHICH OCCURRED
C            IN THAT SUBINTERVAL.  NEXT, FOR EACH SUBINTERVAL THE DIFFERENCE
C            BETWEEN THE MIDPOINT OF THE SUBINTERVAL AND THE DISTRIBUTION
C            MEAN IS EVALUATED AND DIVIDED BY THE STANDARD DEVIATION TO
C            PRODUCE VARIABLE TEM:
C
C                           TEM = (XOUT(I) - AMEAN)/SIGMA
C
C            THEN THE NORMAL DISTRIBUTION FIT FOR THE I-TH INTERVAL IS
C            COMPUTED;
C
C                           TEM = IIMAX*EXP(-0.5*TEM*TEM)
```

```
C
C           AND THE DATA PAIR (XOUT(I), TEM) IS OUTPUT TO THE PLOT FILE.
C
C         B. LOGNORMALLY DISTRIBUTED ALGORITHM RESULTS:
C
C           THE RATIO SSIGMA = SIGMA/AMEAN IS COMPUTED AND THE NATURAL
C           LOGARITHM OF AMEAN IS SET EQUAL TO VARIABLE AMLN.  NEXT THE
C           NATURAL LOGARITHM OF EACH SUBINTERVAL BOUNDARY IS COMPUTED
C           AND STORED IN ARRAY XXLN.  THEN THE SUBINTERVAL CONTAINING
C           THE GREATEST NUMBER OF ALGORITHM RESULTS IS DETERMINED AND
C           INTEGER VARIABLE IIMAX IS SET EQUAL TO THE NUMBER OF ALGORITHM
C           RESULTS WHICH OCCURRED IN THAT SUBINTERVAL.  NEXT A NORMAL
C           CURVE VALUE (TEM1 AND TEM2) IS COMPUTED AT THE BOUNDARY OF
C           EACH SUBINTERVAL;
C
C                       TEM1 = (XXLN(I) - AMLN)/SSIGMA
C                       TEM1 = EXP(-0.5*TEM1*TEM1)
C                       TEM2 = (XXLN(I+1) - AMLN)/SSIGMA
C                       TEM2 = EXP(-0.5*TEM2*TEM2)
C
C           AND THE AVERAGE OF THE TWO RESULTS (TEM) IS DETERMINED,
C           MULTIPLIED BY THE WIDTH OF THE SUBINTERVAL (XXLN(I+1) - XXLN(I)),
C           AND STORED IN ARRAY XXII.  THE RESULTING LOGNORMAL DISTRIBUTION
C           IS THEN SCALED BY IIMAX AND OUTPUT TO THE PLOT FILE.
C
C         C. BETA DISTRIBUTED ALGORITHM RESULTS:
C
C           THE INTEGER VARIABLE IIMAX IS OBTAINED AS ABOVE FOR THE NORMALLY
C           DISTRIBUTED CASE.  NEXT THE COMPUTED MEAN AND STANDARD DEVIATION
C           OF THE RESULTS DISTRIBUTION ARE SUPPLIED TO SUBROUTINE BETAFIT
C           IN WHICH AN APPROPRIATE TABULATED OR CALCULATED BETA DISTRIBUTION,
C           CORRESPONDING TO THE COMPUTED MEAN AND STANDARD DEVIATION, IS
C           GENERATED.  THIS GENERATED CURVE IS THEN SCALED USING THE VALUE
C           OF IIMAX AND THE RESULTING DISTRIBUTION IS OUTPUT TO THE PLOT FILE
C
C         D. UNIFORMLY DISTRIBUTED ALGORITHM RESULTS:
C
C           THE COMPUTED MEAN OF THE ALGORITHM RESULTS DISTRIBUTION IS
C           COMPARED TO THE FULL RANGE (XX(1) TO XX(NINTV+1)) OF THE
C           DISTRIBUTION TO DETERMINE ITS RELATIVE LOCATION IN THE FULL
C           RANGE.  A UNIFORM DISTRIBUTION WITH 50% OF ITS VALUES BETWEEN
C           XX(1) AND AMEAN AND 50% BETWEEN AMEAN AND XX(NINTV+1) IS THEN
C           COMPUTED AND OUTPUT TO THE PLOT FILE.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
```

```
      REAL*4    XX(*), XOUT(*), XXLN(*), XXII(*)
      INTEGER*4 II(*)
C
C      EVALUATE THE SUBINTERVAL MIDPOINTS AND OUTPUT THE HISTOGRAM
C      DATA FOR THE COMPUTED RESULTS DISTRIBUTION TO THE PLOT FILE.
C
      DO 10 I = 1, NINTV
         XOUT(I) = 0.5*(XX(I) + XX(I+1))
         WRITE(NPLT,1000) XOUT(I), II(I)
   10 CONTINUE
C
C      COMPUTE A FIT TO NORMALLY DISTRIBUTED ALGORITHM RESULTS.
C
      IF (ITYPE.EQ.1) THEN
        IIMAX = 0
        DO 100 I = 1, NINTV
           IF (II(I).GT.IIMAX) IIMAX = II(I)
  100   CONTINUE
        DO 110 I = 1, NINTV
           TEM = (XOUT(I) - AMEAN)/SIGMA
           TEM = IIMAX*EXP(-0.5*TEM*TEM)
           WRITE(NPLT,1100) XOUT(I), TEM
  110   CONTINUE
C
C      COMPUTE A FIT TO LOGNORMALLY DISTRIBUTED ALGORITHM RESULTS.
C
      ELSE IF (ITYPE.EQ.2) THEN
        SSIGMA = SIGMA/AMEAN
        AMLN = ALOG(AMEAN)
        DO 200 I = 1, NINTV+1
           XXLN(I) = ALOG(XX(I))
  200   CONTINUE
        IIMAX = 0
        DO 210 I = 1, NINTV
           IF ((XX(I).LT.AMEAN).AND.(XX(I+1).GE.AMEAN)) IIMAX = II(I)
  210   CONTINUE
        DO 220 I = 1, NINTV
           TEM1 = (XXLN(I) - AMLN)/SSIGMA
           TEM1 = EXP(-0.5*TEM1*TEM1)
           TEM2 = (XXLN(I+1) - AMLN)/SSIGMA
           TEM2 = EXP(-0.5*TEM2*TEM2)
           TEM = 0.5*(TEM1 + TEM2)
           XXII(I) = TEM*(XXLN(I+1) - XXLN(I))
  220   CONTINUE
        DO 230 I = 1, NINTV
           IF ((XXLN(I).LT.AMLN).AND.(XXLN(I+1).GE.AMLN))
```

```
     1        SCALE = IIMAX/XXII(I)
  230   CONTINUE
        DO 240 I = 1, NINTV
           TEM = SCALE*XXII(I)
           WRITE(NPLT,1100) XOUT(I), TEM
  240   CONTINUE
C
C      COMPUTE A FIT TO BETA DISTRIBUTED ALGORITHM RESULTS.
C
      ELSE IF (ITYPE.EQ.3) THEN
        IIMAX = 0
        DO 300 I = 1, NINTV
           IF ((XX(I).LT.AMEAN).AND.(XX(I+1).GE.AMEAN)) IIMAX = I
  300   CONTINUE
C
C      CALL BETAFIT TO EVALUATE THE ACTUAL BETA DISTRIBUTION FIT.
C
        CALL BETAFIT(AMEAN, SIGMA, XX, XXLN)
        SCALE = 0.0
        NSCALE = 5
        DO 310 I = IIMAX - NSCALE, IIMAX + NSCALE
           SCALE = SCALE + 2.0*II(I)/(XXLN(I) + XXLN(I+1))
  310   CONTINUE
        SCALE = SCALE/(2*NSCALE + 1)
        DO 320 I = 1, NINTV
           TEM = 0.5*(XXLN(I) + XXLN(I+1))*SCALE
           WRITE(NPLT,1100) XOUT(I), TEM
  320   CONTINUE
C
C      COMPUTE A FIT TO UNIFORMLY DISTRIBUTED ALGORITHM RESULTS.
C
      ELSE IF (ITYPE.EQ.4) THEN
        SCALE = NINTV*(AMEAN - XX(1))/(XX(NINTV+1) - XX(1))
        DO 400 I = 1, NINTV
           IF (XOUT(I).LE.AMEAN) THEN
             TEM = 0.5*ISIZE/SCALE
           ELSE
             TEM = 0.5*ISIZE/(NINTV - SCALE)
           ENDIF
           WRITE(NPLT,1100) XOUT(I), TEM
  400   CONTINUE
      ENDIF
C
C      COMPUTE THE CUMULATIVE PROBABILITY DISTRIBUTION ASSOCIATED
C      WITH THE COMPUTED ALGORITHM RESULTS DISTRIBUTION.
C
```

```
      WRITE(NPLT,1100) XX(1), 0.0
      AREA = 0.0
      DO 410 I = 1, NINTV
         AREA = AREA + (XX(I+1) - XX(I))*II(I)
  410 CONTINUE
      TOTAREA = AREA
      AREA = 0.0
      DO 420 I = 1, NINTV
         AREA = AREA + (XX(I+1) - XX(I))*II(I)
         WRITE(NPLT,1100) XX(I+1), AREA/TOTAREA
  420 CONTINUE
      RETURN
 1000 FORMAT(E12.5,', ',I5)
 1100 FORMAT(E12.5,', ',E12.5)
      END
```

```
C               SIGMA = SIGMA/(XX(NINTV+1) - X(I))
C               TEM   = (AMEDIAN - AMEDIAN*AMEDIAN)/(SIGMA*SIGMA) - 1.
C               ALPHA = AMEDIAN*TEM
C               BETA  = (1.0 - AMEDIAN)*TEM
C
C      ONCE THE VALUES FOR ALPHA AND BETA ARE AVAILABLE, FUNCTION ROUTINE FX
C      IS CALLED TO COMPUTE THE (UNNORMALIZED) BETA DISTRIBUTION, THE AREA
C      UNDER THE DISTRIBUTION IS COMPUTED, AND THEN THE DISTRIBUTION IS
C      NORMALIZED.  THE RESULTING DISTRIBUTION IS RETURNED TO THE CALLING
C      SUBROUTINE IN ARRAY Y.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      REAL*4    AMED(81), ALPH(81), BET(81)
      REAL*4    XX(*), Y(*)
      DATA AMED/0.167,0.172,0.179,0.185,0.192,0.200,0.208,0.217,0.227,
     1          0.238,0.250,0.256,0.263,0.270,0.278,0.286,0.294,0.303,
     2          0.313,0.323,0.333,0.339,0.345,0.351,0.357,0.364,0.370,
     3          0.377,0.385,0.392,0.400,0.408,0.417,0.426,0.435,0.444,
     4          0.455,0.465,0.476,0.488,0.500,0.512,0.524,0.535,0.545,
     5          0.556,0.565,0.574,0.583,0.592,0.600,0.608,0.615,0.623,
     6          0.630,0.636,0.643,0.649,0.655,0.661,0.667,0.677,0.687,
     7          0.697,0.706,0.714,0.722,0.730,0.737,0.744,0.750,0.762,
     8          0.773,0.783,0.792,0.800,0.808,0.815,0.821,0.828,0.833/
      DATA ALPH/1.04, 1.07, 1.11, 1.14, 1.18, 1.23, 1.28, 1.33, 1.39,
     1          1.46, 1.54, 1.58, 1.63, 1.68, 1.73, 1.79, 1.85, 1.92,
     2          1.99, 2.07, 2.16, 2.20, 2.26, 2.31, 2.36, 2.42, 2.48,
     3          2.55, 2.62, 2.69, 2.77, 2.86, 2.95, 3.05, 3.15, 3.26,
     4          3.39, 3.52, 3.66, 3.83, 41*4.0/
      DATA BET/41*4.0, 3.83, 3.66, 3.52, 3.39,
     1          3.26, 3.15, 3.05, 2.95, 2.86, 2.77, 2.69, 2.62, 2.55,
     2          2.48, 2.42, 2.36, 2.31, 2.26, 2.20, 2.16, 2.07, 1.99,
     3          1.92, 1.85, 1.79, 1.73, 1.68, 1.63, 1.58, 1.54, 1.46,
     4          1.39, 1.33, 1.28, 1.23, 1.18, 1.14, 1.11, 1.07, 1.04/
C
C      COMPUTE THE MEDIAN OF THE ALGORITHM RESULTS DISTRIBUTION.
C
      AMEDIAN = (AMEAN - XX(1))/(XX(NINTV + 1) - XX(1))
C
C      DETERMINE ALPHA AND BETA FOR THE TABULATED DISTRIBUTION CASE.
C
      IF (ITABL.EQ.1) THEN
        TEMP1 = 0.5*(AMED(1) + AMED(2))
        TEMP2 = 0.5*(AMED(80) + AMED(81))
        IF (AMEDIAN.LE.TEMP1) THEN
          IBET = 1
```

```
        ELSE IF (AMEDIAN.GT.TEMP2) THEN
          IBET = 81
        ELSE
          DO 10 I = 2, 80
             TEMP1 = 0.5*(AMED(I) + AMED(I-1))
             TEMP2 = 0.5*(AMED(I) + AMED(I+1))
             IF ((AMEDIAN.GT.TEMP1).AND.(AMEDIAN.LE.TEMP2)) IBET = I
   10      CONTINUE
        ENDIF
        ALPHA = ALPH(IBET)
        BETA  = BET(IBET)
C
C       DETERMINE ALPHA AND BETA FOR THE CALCULATED DISTRIBUTION CASE.
C
      ELSE IF (ITABL.EQ.2) THEN
        SIGMA = SIGMA/(XX(NINTV + 1) - XX(1))
        TEM = (AMEDIAN - AMEDIAN*AMEDIAN)/(SIGMA*SIGMA) - 1.0
        ALPHA = AMEDIAN*TEM
        BETA  = (1.0 - AMEDIAN)*TEM
      ENDIF
C
C       INITIALIZE THE POINTS IN THE UNIT INTERVAL AT WHICH THE BETA
C       DISTRIBUTION WILL BE COMPUTED.
C
      Y(1) = 0.0
      Y(NINTV+1) = 1.0
      DO 20 I = 2, NINTV
         Y(I) = (XX(I) - XX(1))/(XX(NINTV + 1) - XX(1))
   20 CONTINUE
C
C       COMPUTE THE UNNORMALIZED BETA DISTRIBUTION AND ITS AREA.
C
      SUMNORM = 0.0
      DO 30 I = 1, NINTV
         FX1 = FX(ALPHA, BETA, Y(I))
         FX2 = FX(ALPHA, BETA, Y(I+1))
         SUMNORM = SUMNORM + 0.5*(FX1 + FX2)*(Y(I+1) - Y(I))
   30 CONTINUE
C
C       COMPUTE THE NORMALIZED BETA DISTRIBUTION.
C
      DO 40 I = 1, NINTV+1
         FX1 = FX(ALPHA, BETA, Y(I))
         Y(I) = FX1/SUMNORM
   40 CONTINUE
      RETURN
```

```
END
```

```
      FUNCTION FX(ALPHA, BETA, YY)
C
C      FUNCTION FX EVALUATES THE FUNCTION:
C
C                  (ALPHA-1.)       (BETA-1.)
C                YY          (1.-YY)
C
C      FOR DELIVERED VALUES OF ALPHA, BETA, AND YY.  THE ABOVE FUNCTION
C      IS THE INTEGRAND IN THE INTEGRAL FORMULA FOR THE BETA
C      PROBABILITY DISTRIBUTION WITH PARAMETERS ALPHA AND BETA.
C
      IF ((YY.GE.1.0).OR.(YY.LE.0.0)) THEN
        FX = 0.0
      ELSE
        IF(((ALPHA.GE.0.9).AND.(ALPHA.LE.1.1)).OR.
     1     ((BETA.GE.0.9).AND.(BETA.LE.1.1))) THEN
          FAC1 = EXP((ALPHA - 1.0)*ALOG(YY))
          FAC2 = EXP((BETA - 1.0)*ALOG(1.0 - YY))
          FX = FAC1*FAC2
        ELSE
          FX = (YY**(ALPHA - 1.0))*(1.0 - YY)**(BETA - 1.0)
        ENDIF
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE OPTIONS
C
C       SUBROUTINE OPTIONS ENABLES THE INTERACTIVE USER OF BETAFACT TO
C       MODIFY THE PROBLEM BEING EXECUTED BY THE PROGRAM.  THE SUBROUTINE
C       LISTS THE AVAILABLE OPTIONS FOR THE USER AND PROMPTS FOR AN
C       OPTION SPECIFICATION.  CONTROL IS THEN RETURNED TO THE MAIN
C       PROGRAM WHICH INTERPRETS THE USER SELECTED OPTION AND PROCEEDS
C       ACCORDINGLY.
C
      COMMON /BLK01/ IPARAM, ITYPE, ISIZE, ISEED, IALGO, IXVAL, ITABL,
     1               IXALG, IOPT, NINTV, NINC, NINP, NOUT, NPLT, NPRT
      WRITE(NOUT,1000)
 1000 FORMAT(/,' SELECT OPTION TO QUIT OR TO RUN A MODIFIED VERSION',/,
     1         ' OF THE CURRENT PROBLEM:',/,/,
     2         '      0   - TERMINATE PROBLEM.',/,
     3         '      1   - CHANGE INDEPENDENT PARAMETER VALUES.',/,
     4         '      2   - CHANGE NUMBER OF ALGORITHM EVALUATIONS',/,
     5         '            AND/OR THE RANDOM NUMBER SEED INTEGER.',/,
     6         '      3   - CHANGE PARAMETER NOMINAL VALUES AND/OR',/,
     7         '            ASSOCIATED K-FACTOR UNCERTAINTIES.',/,
     8         '      4   - CHANGE UNCERTAINTY DISTRIBUTION TYPE.',/,
     9         '      5   - OPTIONS 1 AND 2.',/,
     *         '      6   - OPTIONS 1 AND 3.',/,
     *         '      7   - OPTIONS 1 AND 4.',/,
     *         '      8   - OPTIONS 2 AND 3.',/,
     *         '      9   - OPTIONS 2 AND 4.',/,
     *         '     10   - OPTIONS 3 AND 4.',/,
     *         '     11   - OPTIONS 1, 2, AND 3.',/,
     *         '     12   - OPTIONS 1, 2, AND 4.',/,
     *         '     13   - OPTIONS 1, 3, AND 4.',/,
     *         '     14   - OPTIONS 2, 3, AND 4.',/,
     *         '     15   - OPTIONS 1, 2, 3, AND 4.',/,
     *         '     16   - COMPLETELY MODIFY THE CURRENT PROBLEM.',/)
      READ(NINP,*) IOPT
      RETURN
      END
```

```
      SUBROUTINE FNRN(NUMNR,RN)
C
C      SUBROUTINE FNRN RETURNS A NORMALLY DISTRIBUTED RANDOM NUMBER
C      WITH VARIABLE NAME RN.  THE INTEGER NUMNR IS USED ONLY DURING
C      THE FIRST CALL TO THE SUBROUTINE AND ITS VALUE IS NOT CRITICAL.
C      HOWEVER, IF A DIFFERENT STRING OF RANDOM NUMBERS IS DESIRED
C      EITHER DURING THE COURSE OF A SINGLE BETAFACT ANALYSIS OR FROM
C      ONE ANALYSIS TO ANOTHER, THE VALUE FOR NUMNR MUST BE CHANGED
C      AT THE BEGINNING OF EACH ANALYSIS.  THE SUBROUTINE CALLS
C      FUNCTION GASDEV WHICH COMPUTES A VALUE FOR RN.
C
      DATA INFIRST /0/
      IF (INFIRST.EQ.0) THEN
        INFIRST = 1
        IF (NUMNR.GE.0) THEN
          IDUM = -NUMNR
        ELSE
          IDUM = NUMNR
        ENDIF
      ENDIF
      RN = GASDEV(IDUM)
      RETURN
      END
```

```
      SUBROUTINE UPR1(NUMUR,RN)
C
C      SUBROUTINE UPR1 RETURNS A UNIFORMLY DISTRIBUTED RANDOM NUMBER
C      WITH VARIABLE NAME RN.  THE INTEGER NUMUR IS USED ONLY DURING
C      THE FIRST CALL TO THE SUBROUTINE AND ITS VALUE IS NOT CRITICAL.
C      HOWEVER, IF A DIFFERENT STRING OF RANDOM NUMBERS IS DESIRED
C      EITHER DURING THE COURSE OF A SINGLE BETAFACT ANALYSIS OR FROM
C      ONE ANALYSIS TO ANOTHER, THE VALUE FOR NUMUR MUST BE CHANGED
C      AT THE BEGINNING OF EACH ANALYSIS.  THE SUBROUTINE CALLS
C      FUNCTION RAN1 WHICH COMPUTES A VALUE FOR RN.
C
      DATA IUFIRST /0/
      IF (IUFIRST.EQ.0) THEN
        IUFIRST = 1
        IF (NUMUR.GE.0) THEN
          IDUM = -NUMUR
        ELSE
          IDUM = NUMUR
        ENDIF
      ENDIF
      RN = RAN1(IDUM)
      RETURN
      END
```

```
      FUNCTION GASDEV(IDUM)
C
C      THE FOLLOWING ROUTINE IS TAKEN FROM:
C       NUMERICAL RECIPES - THE ART OF SCIENTIFIC COMPUTING
C       W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND
C       W. T. VETTERLING; CAMBRIDGE UNIVERSITY PRESS 1988.
C       P. 202-203.
C
C      RETURNS A NORMALLY DISTRIBUTED DEVIATE WITH ZERO MEAN AND
C      UNIT VARIANCE, USING RAN1(IDUM) AS THE SOURCE OF UNIFORM
C      DEVIATES.
C
      DATA ISET /0/
      IF (ISET.EQ.0) THEN
    1   V1 = 2.*RAN1(IDUM) - 1.
        V2 = 2.*RAN1(IDUM) - 1.
        RRR = V1**2 + V2**2
        IF (RRR.GE.1.) GO TO 1
        FAC = SQRT(-2.*ALOG(RRR)/RRR)
        GSET = V1*FAC
        GASDEV = V2*FAC
        ISET = 1
      ELSE
        GASDEV = GSET
        ISET = 0
      ENDIF
      RETURN
      END
```

```
      FUNCTION RAN1(IDUM)
C
C      THE FOLLOWING ROUTINE IS TAKEN FROM:
C       NUMERICAL RECIPES - THE ART OF SCIENTIFIC COMPUTING
C       W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND
C       W. T. VETTERLING; CAMBRIDGE UNIVERSITY PRESS 1988.
C       P. 196-197
C
C      RETURNS A UNIFORM RANDOM DEVIATE BETWEEN 0.0 AND 1.0.
C      SET IDUM TO ANY NEGATIVE VALUE TO INITIALIZE OR
C      REINITIALIZE THE SEQUENCE.
C
C      CONSTANTS ARE FOR COMPUTERS WHICH OVERFLOW AT POSITIVE
C      INTEGER VALUES OF 2**31 - 1.
C
      DIMENSION RR(97)
      PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=1./M1)
      PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=1./M2)
      PARAMETER (M3=243000,IA3=4561,IC3=51349)
      DATA IFF /0/
      IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
        IFF = 1
        IX1 = MOD(IC1-IDUM,M1)
        IX1 = MOD(IA1*IX1+IC1,M1)
        IX2 = MOD(IX1,M2)
        IX1 = MOD(IA1*IX1+IC1,M1)
        IX3 = MOD(IX1,M3)
        DO 11 J = 1, 97
          IX1 = MOD(IA1*IX1+IC1,M1)
          IX2 = MOD(IA2*IX2+IC2,M2)
          RR(J) = (FLOAT(IX1) + FLOAT(IX2)*RM2)*RM1
   11   CONTINUE
        IDUM = 1
      ENDIF
      IX1 = MOD(IA1*IX1+IC1,M1)
      IX2 = MOD(IA2*IX2+IC2,M2)
      IX3 = MOD(IA3*IX3+IC3,M3)
      J = 1 + (97*IX3)/M3
      IF (J.GT.97.OR.J.LT.1) PAUSE
      RAN1 = RR(J)
      RR(J) = (FLOAT(IX1) + FLOAT(IX2)*RM2)*RM1
      RETURN
      END
```

DISTRIBUTION LIST

DNA-TR-90-81

DEPARTMENT OF DEFENSE

DEFENSE ADVANCED RSCH PROJ AGENCY
ATTN: F PATTEN

DEFENSE INTELLIGENCE AGENCY
ATTN: RTS-2B

DEFENSE NUCLEAR AGENCY
ATTN: SPSP (CAPT LAWRY)
4 CYS ATTN: TITL

DEFENSE TECHNICAL INFORMATION CENTER
2 CYS ATTN: DTIC/FDAB

STRATEGIC DEFENSE INITIATIVE ORGANIZATION
ATTN: TNK LT COL MARTIN

UNDER SECRETARY OF DEFENSE
ATTN: ENGR TECH J PERSCH

DEPARTMENT OF THE NAVY

OFFICE OF NAVAL TECHNOLOGY
ATTN: CODE 217

DEPARTMENT OF THE AIR FORCE

AIR FORCE CTR FOR STUDIES & ANALYSIS
ATTN: AFCSA/SAMI

FOREIGN TECHNOLOGY DIVISION
ATTN: FTD/TTX-1/JOHN A TUSS

STRATEGIC AIR COMMAND/XRFS
ATTN: XRFS

WRIGHT RESEARCH & DEVELOPMENT CENTER
ATTN: MLP

DEPARTMENT OF ENERGY

LAWRENCE LIVERMORE NATIONAL LAB
ATTN: F SERDUKE
ATTN: M GERASSIMENKO

LOS ALAMOS NATIONAL LABORATORY
ATTN: X-1 E531, R S DINGUS

SANDIA NATIONAL LABORATORIES
ATTN: DIV 8242 (M BIRNBAUM)

DEPARTMENT OF DEFENSE CONTRACTORS

APTEK, INC
2 CYS ATTN: S H SUTHERLAND
ATTN: VICE PRESIDENT-ENGINEERING

KAMAN SCIENCES CORP
ATTN: D MOFFETT
ATTN: DASIAC
ATTN: M BROSEE
ATTN: R ALMASSY

KAMAN SCIENCES CORPORATION
ATTN: DASIAC

MCDONNELL DOUGLAS CORPORATION
ATTN: J S KIRBY

PHYSICAL SCIENCES, INC
ATTN: DR PETER WU

R & D ASSOCIATES
ATTN: D GAKENHEIMER

RAND CORP
ATTN: E HARRIS

S-CUBED
ATTN: G GURTMAN

SPARTA, INC
ATTN: J E LOWDER
ATTN: R G ROOT

SRI INTERNATIONAL
ATTN: B HOLMES